Universidade de Aveiro Departamento de Matemática
2008

António Ferreira Dias

O problema da p-mediana aplicado ao problema da gestão óptima da diversidade.

Universidade de Aveiro Departamento de Matemática
2008

António Ferreira Dias

O problema da p-mediana aplicado ao problema da gestão óptima da diversidade

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática e Aplicações, realizada sob a orientação científica da Dr.ª Gladys Castillo Jordán e do Dr. Agostinho Miguel Mendes Agra, Professores Auxiliares do Departamento de Matemática da Universidade de Aveiro.

Dedico este trabalho à minha esposa e filhas pelo incansável apoio.

o júri

Presidente

Prof. Dr. Domingos Moreira Cardoso
professor catedrático da Universidade de Aveiro

Prof.ª Dr.ª Maria da Conceição da Fonseca
professora auxiliar da Faculdade de Ciências da Universidade de Lisboa

Prof.ª Dr.ª Gladys Castillo Jordán
professora auxiliar da Universidade de Aveiro

Prof. Dr. Agostinho Miguel Mendes Agra
professor auxiliar da Universidade de Aveiro

agradecimentos

Aos meus orientadores, Dr.ª Gladys Castillo Jordán e Dr. Agostinho Miguel Mendes Agra, pela disponibilidade e grande apoio.

palavras-chave

Problema da p-mediana, metaheurísticas, problema da gestão óptima da diversidade.

resumo

Neste trabalho aborda-se o problema da p-mediana e a sua aplicação a um problema da gestão óptima da diversidade na indústria automóvel. O problema da p-mediana é aplicado em diversas situações práticas reais, nomeadamente na localização de equipamentos públicos, industriais, comerciais e de telecomunicações. Na sua forma geral o problema da p-mediana é NP-difícil e na sua resolução são implementados métodos heurísticos. Primeiramente é feita uma apresentação do problema, e são descritos os principais métodos heurísticos de resolução. De seguida, é descrito o problema da gestão óptima da diversidade e apresentada uma aplicação deste problema à indústria automóvel. Por último, é efectuado um estudo computacional de modo a avaliar o desempenho de diferentes versões do algoritmo híbrido e, simultaneamente, comparar essas versões com o algoritmo guloso.

keywords P-median problem, metaheuristics, optimal diversity management problem.

abstract This work presents the p-median problem and its application to an optimal diversity management problem in the automobile industry. The p-median problem is applied in different real practical situations, particularly in public, industrial, commercial and telecommunications equipments location. In its general form, it's a NP-hard problem and in its resolution heuristic methods are implemented. First, there is a presentation of the problem, and the main methods of heuristic resolution are described. Next it is described the optimal diversity management problem and an application of this problem is submitted to the car industry. Finally, it is reported a computational study to assess the performance of different versions of hybrid algorithm and compare these versions simultaneously with the greedy algorithm.

Índice

Lista de figuras

Lista de tabelas

1. Introdução

O problema da p-mediana faz parte de uma classe mais ampla de problemas, conhecidos como problemas de localização, que tratam de decisões sobre onde localizar equipamentos, considerando um conjunto de clientes que devem ser servidos de forma a optimizar um determinado critério.

Nesta tese é efectuada uma abordagem ao problema da p-mediana que, em linhas gerais, pode ser definido como se segue. São dados: um grafo, um conjunto de potenciais localizações dos equipamentos, um conjunto de clientes (ou vértices de procura), as distâncias percorridas (ou custos incorridos) para satisfazer a procura de cada cliente a partir de cada equipamento, e uma constante p representando o número de equipamentos a localizar. Objectivo: localizar p equipamentos nos vértices do grafo de modo a minimizar a soma das distâncias (ou custos de transporte) de cada cliente ao equipamento que lhe fica mais próximo.

Neste problema, o benefício (custo) associado a cada par cliente/equipamento diminui (aumenta) gradualmente com a distância entre o cliente e o equipamento. Esta distância e o custo associado ao par cliente/equipamento estão, normalmente, relacionados linearmente. Por exemplo, o custo de servir uma loja a partir de um armazém depende do tempo dispendido na deslocação do armazém para a loja. Neste caso, a relação entre o custo e a distância entre o armazém e a loja é aproximadamente linear.

São diversas as situações práticas reais em que se pretende localizar equipamentos de modo a minimizar a média das distâncias ponderadas entre cada cliente e o equipamento localizado mais próximo. Por exemplo, localização de serviços públicos (escolas, hospitais, bibliotecas), de paragens de autocarro, de armazéns, de antenas de telecomunicações, entre outras. A título de exemplo prático, suponha-se que se pretende localizar bibliotecas numa cidade. Para encontrar boas localizações para as bibliotecas, deve dividir-se a cidade em blocos. A população de cada bloco constitui a procura desse bloco, assumindo-se que essa procura está concentrada no centróide (ou centro geométrico) do bloco. Minimizando a distância entre os centróides e o mais próximo dos p equipamentos – o que se consegue

resolvendo o problema da p-mediana – encontram-se as localizações das *p* bibliotecas que minimizam a média das distâncias que os habitantes da cidade terão de percorrer se se deslocarem de suas casas para a biblioteca mais próxima.

Entre as possíveis aplicações do problema da p-mediana encontra-se o problema da gestão óptima da diversidade estudado nesta tese. Em algumas indústrias, onde o número de configurações que o produto fabricado pode tomar é muito grande, pode haver necessidade de, em vez de todas, produzir apenas um certo número de configurações possíveis. É o caso, por exemplo, das cablagens para a indústria automóvel. A substituição de uma dada configuração por outra mais completa (contendo fios que não vão ser utilizados) representa um elevado custo adicional para as empresas. O problema da gestão óptima da diversidade consiste na determinação de um conjunto óptimo de configurações a produzir, sendo qualquer configuração não produzida substituída pela mais barata, produzida, compatível com ela. O objectivo é minimizar o custo adicional de produção, ou seja, o custo total associado aos fios que não vão ser utilizados. O problema da gestão óptima da diversidade pode ser modelado como um problema da p-mediana num grafo orientado. Neste problema, cada configuração corresponde a um cliente, as configurações a produzir correspondem aos equipamentos e uma configuração é afecta a uma configuração produzida se for substituída por ela.

O problema da p-mediana é um problema de optimização combinatória classificado como NP-difícil [12]. Apesar de o conjunto de soluções possíveis do problema ser finito, e, portanto, poder obter-se a solução óptima por simples enumeração e avaliação de cada possibilidade, o conjunto de soluções possíveis pode ser tão vasto que se torna impossível avaliar cada um dos seus elementos num espaço de tempo aceitável. Boas soluções podem requerer tempos computacionais excessivos para que possam ser consideradas, por exemplo, no contexto de tomadas de decisão. É, por isso, necessária uma abordagem heurística (ou aproximada) para a resolução do problema.

Neste trabalho, são descritos alguns dos principais métodos heurísticos utilizados na resolução do problema da p-mediana, nomeadamente *algoritmos construtivos* (p. ex.: algoritmo guloso [24, 39]; algoritmos gulosos aleatorizados [34]), *procedimentos de pesquisa local* (p. ex.: pesquisa em vizinhança [26]; pesquisa baseado na substituição de vértices [18, 33, 37, 39]) e *metaheurísticas* (p. ex.: pesquisa tabu [28]; pesquisa em vizinhança variável [11, 18, 21]; algoritmos genéticos [7, 9, 23]; GRASP [10]).

É dado um destaque especial à descrição de um algoritmo híbrido proposto por Resende e Werneck em [34], o qual combina elementos de várias metaheurísticas. À semelhança do GRASP, esta heurística é uma abordagem multi-arranque, onde cada iteração consiste na construção aleatorizada de uma solução seguida de pesquisa local. À pesquisa tabu, este método foi buscar a ideia de *religação por caminhos* [25], e aos algoritmos genéticos o conceito de *múltiplas gerações*.

Finalmente, é apresentado um estudo computacional que permite avaliar:

i) o desempenho de diferentes versões do algoritmo híbrido proposto por Resende e Werneck em [34] e do algoritmo guloso, implementado em [1], em termos de qualidade das soluções obtidas e dos tempos de processamento;

ii) o balanço entre a qualidade das soluções obtidas e o custo computacional (tempos de processamento), para cada uma das versões do algoritmo híbrido, tomando como referência os resultados apresentados pelo algoritmo guloso para as instâncias do problema da gestão óptima da diversidade testadas.

Os capítulos subsequentes encontram-se organizados da seguinte forma:

No capítulo 2 é apresentado um breve historial relativo ao estudo do problema da p-mediana, bem como a definição e uma formulação de programação linear do problema.

O capítulo 3 é dedicado à descrição dos métodos heurísticos mais conhecidos e mais utilizados na resolução de problemas de optimização combinatória. São eles: o algoritmo construtivo guloso, as heurísticas de melhoramento pesquisa em vizinhança e pesquisa baseada na substituição de vértices, e as metaheurísticas pesquisa tabu, pesquisa em vizinhança variável, algoritmos genéticos e GRASP (procedimento de pesquisa guloso aleatorizado e adaptativo).

Em seguida, no capítulo 4, é apresentado o algoritmo híbrido para o problema da p-mediana proposto por Resende e Werneck em [34].

O capítulo 5 introduz o problema da gestão óptima da diversidade, e apresenta a sua aplicação a um problema real da indústria automóvel: o problema da diversidade e distribuição de cablagens.

É apresentado, no capítulo 6, um estudo computacional cujo principal objectivo é comparar o algoritmo guloso com três versões do algoritmo híbrido relativamente ao balanço entre a qualidade das soluções obtidas e os tempos de processamento, quando aplicados a instâncias do problema da gestão óptima da diversidade.

Por último, o capítulo 7 mostrará as conclusões sobre o trabalho realizado.

2. O Problema da p-Mediana

Neste capítulo, é apresentado um breve historial do problema da p-mediana, seguido da sua definição e formulação matemática como problema de programação linear inteira.

2.1. Historial e Desenvolvimento

O problema da p-mediana encontra pontos (designados medianas) num dado conjunto finito de pontos, de modo a minimizar a distância média ou total entre pontos e medianas. Este tipo de problema teve a sua origem no século XVII, quando Pierre de Fermat colocou a seguinte questão: dados os vértices de um triângulo (três pontos no plano), encontrar o ponto do plano (mediana), tal que a soma das distâncias entre cada um dos vértices e a mediana seja mínima [30]. No início do século XX, Alfred Weber apresentou o mesmo problema, com a atribuição de pesos a cada um dos três pontos para simular pedidos de clientes [30]. Nesta situação, cada um dos três pontos corresponde a um cliente, os pesos às procuras dos clientes, e encontrar a mediana corresponde a encontrar a melhor localização de um equipamento para satisfazer os pedidos dos três clientes. Este problema é usualmente apresentado como o primeiro problema de localização. Mais tarde, este problema foi generalizado quer para encontrar a mediana de mais de três pontos no plano quer para seleccionar $p > 1$ medianas com localização contínua no plano (problema de Weber com vários equipamentos).

O problema de Weber localiza medianas (equipamentos) em localizações contínuas no plano Euclidiano. No início dos anos 60, Hakimi desenvolveu problemas similares aos de Weber para encontrar medianas num grafo [17]. No seu problema da mediana absoluta, que é semelhante ao problema ponderado de Weber, Hakimi definiu *mediana absoluta* como o ponto, num grafo, que minimiza a soma das distâncias ponderadas entre esse ponto e os vértices do grafo. Hakimi permitiu que a mediana absoluta se situasse em qualquer lugar ao longo das arestas do grafo, mas provou que uma mediana absoluta óptima está,

sempre, localizada num vértice do grafo. Em [17], Hakimi generaliza o problema da mediana absoluta para encontrar p medianas num grafo e, mais uma vez, mostrou que, embora nem todas as soluções óptimas deste problema tenham as medianas localizadas apenas em vértices, existe sempre um conjunto de p vértices que minimiza o objectivo. Deste modo, Hakimi proporciona uma representação discreta de um problema contínuo, restringindo a procura da solução do problema aos vértices do grafo. O problema da p-mediana difere do problema de Weber porque é discreto, uma consequência de se restringirem as localizações possíveis das medianas ao conjunto de vértices. Também é definido num grafo, e não no plano como no problema de Weber, no qual as distâncias são definidas pela métrica Euclidiana.

Hakimi desenvolveu os problemas da mediana absoluta e da p-mediana com o objectivo de encontrar a melhor localização para a instalação de centro(s) de comutação, numa rede de comunicação. O sucesso obtido por Hakimi fez com que, desde então, o problema da p-mediana tenha sido inseparável da teoria da localização, tornando-se um dos mais comuns modelos de localização de equipamentos. O problema da p-mediana é, assim, usado para modelar diversas situações reais, tais como a localização de instalações industriais, de armazéns e de equipamentos públicos.

2.2. Definições

Considere-se um grafo $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ onde $\mathbf{V} = \{v_1, \ldots, v_n\}$ é o conjunto dos n vértices e $\mathbf{A}$ é o conjunto das arestas. Cada vértice é, simultaneamente, uma possível localização de um equipamento (ou mediana) e um cliente (ou vértice de procura).

Para $i = 1, \ldots, n,$ seja $w_i \geq 0$ o peso (ou procura) atribuído ao vértice (cliente) v_i. Define-se a menor distância do vértice v_i ao vértice v_j, que se representa por $d(v_i, v_j)$, como o comprimento do menor caminho entre os vértices v_i e v_j.

A menor distância ponderada entre o vértice de procura v_i e o equipamento v_j (ou custo decorrente da satisfação da procura do cliente v_i pelo equipamento v_j) é definida como $w_i d(v_i, v_j)$.

O problema da p-mediana consiste em encontrar um conjunto de vértices (localizações de equipamentos) $\mathbf{V}_p \subseteq \mathbf{V}$ tal que $\left| \mathbf{V}_p \right| = p$, de modo a minimizar a soma das

menores distâncias ponderadas entre os vértices de procura em $\mathbf{V}\backslash\mathbf{V}_p$ e o equipamento mais próximo em $\mathbf{V}_p$.

Para qualquer conjunto de p vértices $\mathbf{X}_p \subseteq \mathbf{V}$ e para qualquer vértice $v \in \mathbf{V}$, define-se a menor distância do vértice v ao elemento mais próximo em $\mathbf{X}_p$ como $d(v,\mathbf{X}_p)=\min\{d(v, x) \mid x\in\mathbf{X}_p\}$.

A solução óptima do problema da p-mediana é o conjunto de p vértices, $\mathbf{V}_p$, tal que para qualquer $\mathbf{X}_p$,

$$\sum_{i=1}^{n} w_i\, d(v_i, \mathbf{V}_p) \le \sum_{i=1}^{n} w_i\, d(v_i, \mathbf{X}_p).$$

Para cada vértice v_j, em $\mathbf{V}_p$, existe um conjunto de vértices que estão mais próximos desse vértice do que de qualquer outro de $\mathbf{V}_p$, o que origina uma partição do conjunto $\mathbf{V}$ em p células. A célula $\mathbf{P}_j$ da partição é:

$\mathbf{P}_j=\{v_i\in\mathbf{V} \mid d(v_i, v_j) \le d(v_i, v_k),\ \forall\, v_k \in\mathbf{V}_p\backslash\{v_j\}\}$.

Em caso de empate, ou seja, se $d(v_i, v_j) = d(v_i, v_k)$ e $j \neq k$, o vértice v_i é usualmente atribuído ao vértice, em $\mathbf{V}_p$, com o menor índice.

2.3. Formulação Matemática e Propriedades

O problema da p-mediana pode ser formulado em termos de programação inteira binária [35]. Para $i, j \in\{1, \ldots, n\}$, com $i \neq j$, seja x_{ij} uma variável de afectação tal que $x_{ij} = 1$, se o cliente v_i é afecto ao equipamento v_j, e $x_{ij} = 0$, caso contrário. Se no vértice v_j é localizado um equipamento x_{jj} =1, caso contrário $x_{jj} = 0$. Então o problema da p-mediana pode ser formulado do seguinte modo:

Minimizar $$c = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i\, d(v_i, v_j)x_{ij} \qquad (2.1)$$

sujeito a: $$\sum_{j=1}^{n} x_{ij} = 1\ ,\ \forall i = 1, \ldots, n \qquad (2.2)$$

$$\sum_{j=1}^{n} x_{jj} = p \qquad (2.3)$$

$$x_{jj} \ge x_{ij}\ ,\ \forall i, j = 1,\ ...,\ n \qquad (2.4)$$

$$x_{ij} \in\{0, 1\}\ ,\ \forall i, j = 1, \ldots, n \qquad (2.5)$$

A função objectivo (2.1), a minimizar, representa a soma das distâncias ponderadas entre cada cliente e o equipamento mais próximo. A restrição (2.2) assegura que cada cliente v_i seja afecto a um e um só equipamento v_j. A restrição (2.3) garante a localização de exactamente p equipamentos. Portanto, $\mathbf{V}_p = \{v_j \mid x_{jj}=1\}$. A restrição (2.4) impõe que nenhum cliente seja afecto a um vértice onde não esteja localizado um equipamento. A restrição (2.5) impõe a condição de integralidade das variáveis.

A formulação do problema da p-mediana apresentada considera que os equipamentos são localizadas em vértices de um grafo, o que, noutros problemas de localização, onde os equipamentos podem ser também instalados nas arestas, poderia implicar que as soluções encontradas fossem suboptimais. No entanto, como já foi referido, Hakimi mostrou, em 1965, que, para o problema da p-mediana, pelo menos uma solução óptima consiste na localização das p medianas (equipamentos) em vértices do grafo.

Para provar a veracidade desta afirmação, considere-se uma solução em que pelo menos um equipamento é localizado numa aresta (v_i, v_j) a uma distância α do vértice v_i $[0 \leq \alpha \leq d(v_i, v_j)]$. Seja W_i a procura total que é servida por este equipamento, situado na aresta (v_i, v_j), através do vértice v_i. Defina-se W_j, de forma semelhante, com respeito ao vértice v_j. Assuma-se (sem perda de generalidade) que $W_i \geq W_j$. Movendo o equipamento da sua posição para o vértice v_i, que se encontra a uma distância α (sem alteração da afectação de qualquer vértice de procura), provoca-se uma variação de $(W_j - W_i)\alpha$ no valor da função objectivo. Uma vez que $W_i \geq W_j$, esta quantidade é não-positiva. Assim, fazendo esta alteração na localização do equipamento não se aumenta o valor da função objectivo. Isto é suficiente para provar que pelo menos uma solução óptima consiste em localizar os equipamentos apenas em vértices do grafo.

A propriedade anterior limita o número de soluções alternativas a examinar, para encontrar a solução óptima de um problema da p-mediana, a $\binom{n}{p} = \frac{n!}{p!(n-p)!}$, onde n é o número de vértices do grafo e p é o número de equipamentos a localizar. Assim, é possível a resolução de um problema da p-mediana por enumeração e avaliação de todas as suas soluções admissíveis.

Para ilustrar a resolução de um problema da p-mediana através da enumeração e avaliação de todas as soluções admissíveis, considere-se a rede apresentada na figura 2.1

onde se pretende localizar dois equipamentos. Os números nos rectângulos que se encontram junto dos vértices representam as suas procuras.

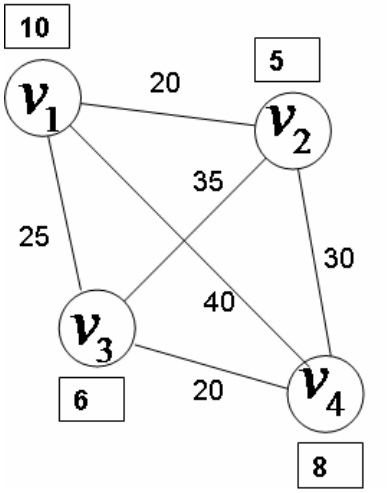

Figura 2.1: Exemplo de uma pequena rede para o problema da p-mediana

A solução óptima do problema pode ser obtida a partir da enumeração e avaliação de cada uma das seis soluções admissíveis. A tabela 2.1 mostra o valor da função objectivo para cada uma das soluções admissíveis.

O custo total mínimo, 220, é obtido quando os equipamentos são localizados nos vértices v_1 e v_4, sendo o cliente v_2 afecto ao equipamento v_1 e o cliente v_3 afecto ao equipamento v_4.

A avaliação de todas as soluções admissíveis de um problema da p-mediana nem sempre é tão fácil como no exemplo apresentado. Por exemplo, se se pretende localizar *10* equipamentos numa rede com *50* vértices, o número de soluções a avaliar é $\binom{50}{10} > 10^{10}$. Se fosse possível avaliar um milhão de combinações por segundo, a avaliação de todas as soluções admissíveis do problema demoraria quase três horas.

Note-se que a avaliação de cada combinação implica encontrar o equipamento mais próximo de cada um dos *40* vértices em que o equipamento não está localizado. Assim, avaliar cada solução requer pelo menos 400 comparações, 40 multiplicações e 40 adições.

Localização dos equipamentos	**Afectação de clientes a equipamentos**	**Valor da função objectivo (Custo)**
v_1 e v_2	v_1 afecto a v_1 v_2 afecto a v_2 v_3 afecto a v_1 v_4 afecto a v_2	390 $(=6\times25+8\times30)$
v_1 e v_3	v_1 afecto a v_1 v_2 afecto a v_1 v_3 afecto v_3 v_4 afecto a v_3	260 $(=5\times20+8\times20)$
v_1 e v_4	v_1 afecto a v_1 v_2 afecto a v_1 v_3 afecto a v_4 v_4 afecto a v_4	220 $(=5\times20+6\times20)$
v_2 e v_3	v_1 afecto a v_2 v_2 afecto a v_2 v_3 afecto a v_3 v_4 afecto a v_3	360 $(=8\times20+10\times20)$
v_2 e v_4	v_1 afecto a v_2 v_2 afecto a v_2 v_3 afecto a v_4 v_4 afecto a v_4	320 $(=10\times20+6\times20)$
v_3 e v_4	v_1 afecto a v_3 v_2 afecto a v_4 v_3 afecto a v_3 v_4 afecto a v_4	400 $(=10\times25+5\times30)$

Tabela 2.1: Valor da função objectivo para cada uma das soluções admissíveis do problema.

3. Algoritmos Heurísticos para o Problema da p-Mediana

O problema da p-mediana é um problema de optimização combinatória, uma vez que o objectivo é seleccionar de um conjunto discreto e finito de dados (vértices), **V**, o subconjunto $\mathbf{V}_p$ que minimiza uma função objectivo. Encontrar a solução óptima de um problema deste género é teoricamente possível por pesquisa exaustiva, isto é, pela enumeração e avaliação de todas as possíveis soluções (combinações). Contudo, na prática, é inviável seguir essa estratégia uma vez que o número de combinações possíveis aumenta exponencialmente com o tamanho do problema. Como provado por Garey e Johnson em [12], o problema da p-mediana é NP-difícil.

Ao longo das últimas décadas muito trabalho tem sido feito no desenvolvimento de métodos de procura da solução óptima que não exijam a avaliação de todas as soluções admissíveis. No entanto, os algoritmos exactos, que garantem a solução óptima deste tipo de problema, têm frequentemente complexidade muito elevada, o que impede a sua aplicação na prática. Nestes casos, empregam-se, usualmente, métodos heurísticos (ou de aproximação) que são procedimentos simples, muitas vezes empíricos, e que produzem boas soluções (não necessariamente a óptima) num tempo computacional razoável. O grande inconveniente resultante da utilização deste tipo de método reside no facto de não ser possível conhecer à priori a qualidade da solução com eles obtida, desconhecendo-se, portanto, quão próximo do óptimo global é a solução encontrada [30].

Neste capítulo são apresentados três tipos de algoritmos heurísticos utilizados na resolução do problema da p-mediana: *heurísticas construtivas*, *heurísticas de pesquisa local* e *metaheurísticas*.

3.1. Heurísticas Construtivas

Heurísticas construtivas são algoritmos que *iterativamente* constroem uma solução admissível do problema. O esquema de um *algoritmo iterativo* para um problema de optimização, em geral, é muito simples: a partir de uma solução inicial admissível

construir outra melhor (com menor valor da função objectivo se o problema é de minimização) e repetir este processo de melhora até verificar um critério de paragem. Seja $\mathbf{X}^0$ a solução inicial admissível do problema de optimização. Na k-ésima iteração, a partir de $\mathbf{X}^{k-1}$ constrói-se uma solução $\mathbf{X}^k$ tal que $c(\mathbf{X}^k) \leq c(\mathbf{X}^{k-1})$, onde $c(.)$ é a função objectivo do problema de minimização. O processo repete-se até se verificar um critério de paragem dado [5].

Os métodos construtivos são um caso particular dos métodos iterativos. Em vez de partir de uma solução admissível e melhorá-la em cada iteração, os métodos construtivos, em geral, partem de uma solução parcial vazia ($\mathbf{X}^0 = \varnothing$) e constroem passo a passo uma solução admissível. Na k-ésima iteração é seleccionado, e inserido na solução parcial $\mathbf{X}^{k-1}$, o melhor *elemento candidato* de forma a construir uma solução $\mathbf{X}^k$. O processo repete-se até ser gerada uma solução admissível próxima da solução óptima. Estes procedimentos construtivos podem ser utilizados isoladamente, contudo, são frequentemente combinados com métodos mais elaborados, como algoritmos de pesquisa local ou metaheurísticas.

3.1.1. Algoritmos Gulosos

Os algoritmos gulosos (*greedy*) são heurísticas que constroem uma solução iterativamente, incorporando, em cada iteração, o melhor elemento candidato na solução parcial. Para determinar o melhor elemento a inserir na solução parcial corrente faz-se uso de uma *função gulosa*, que mede de forma gulosa o benefício de adicionar cada um dos elementos candidatos à solução parcial. Esta medida é gulosa no sentido de que no momento da selecção do "melhor" elemento apenas analisa o benefício para a iteração corrente, não considerando o que pode acontecer em iterações futuras.

Um algoritmo guloso para o problema da p-mediana foi proposto pela primeira vez por Kuehn e Hamburger, em 1963 [24]. No contexto deste problema, uma solução parcial representa um conjunto de localizações de equipamentos e qualquer localização que não pertença à solução considera-se um elemento candidato. O algoritmo começa com uma solução parcial vazia ($\mathbf{X}^0 = \varnothing$) e vai adicionando uma localização de equipamento de cada vez até que p localizações de equipamentos tenham sido seleccionadas. Suponha-se que a solução parcial $\mathbf{X}^k$ inclui as localizações de k equipamentos dos p a localizar, e pretende-se seleccionar a localização do equipamento seguinte para fazer parte da nova solução $\mathbf{X}^{k+1}$.

Denotemos por $d(v_i, \mathbf{X}^k)$ a menor distância entre o vértice de procura v_i e o equipamento mais próximo pertencente ao conjunto $\mathbf{X}^k$. Do mesmo modo, seja $d(v_i, \mathbf{X}^k \cup \{ v_j \})$ a menor distância entre o vértice de procura $v_i \in \mathbf{V}\backslash(\mathbf{X}^k \cup \{v_j\})$ e o elemento do conjunto $\mathbf{X}^k \cup \{v_j\}$ mais próximo de v_i. O melhor lugar para localizar o novo equipamento é o vértice candidato v_j que minimiza a função gulosa $c(v_j, \mathbf{X}^k)$, definida por

$$c\left(v_j, \mathrm{X}^k\right) \equiv \sum_{i \in \left\{l \in \mathbb{N} \mid v_l \in \mathbf{V}\backslash\left(\mathrm{X}^k \cup \{v_j\}\right)\right\}} w_i d\left(v_i, \mathrm{X}^k \cup \{v_j\}\right) \tag{3.1}$$

Na figura 3.1 é apresentado o pseudo-código do algoritmo guloso que pode ser utilizado para resolver o problema da p-mediana.

```
procedimento Guloso(V, p, c(., .)) {
    X ← ∅
    enquanto (|X| < p) {
      calcular c(v_i, X), ∀ v_i ∈ V\X
      v* ← argmin{ c(v_i, X) | v_i ∈ V\X}
      X ← X ∪ {v*}
    }
 retornar X
}
```

Figura 3.1:Pseudo-código do algoritmo guloso para o problema da p-mediana.

Para ilustrar este método, considere-se a rede apresentada na figura 3.2. Os números contidos nos rectângulos junto aos vértices são as procuras, w_i.

A tabela A1 (anexo A) apresenta a matriz de distâncias. As distâncias ponderadas $w_i d(v_i, v_j)$ são mostrados na tabela A2 (anexo A).

Somando as entradas em cada coluna da tabela A2, obtêm-se os custos (valores da função gulosa) para cada um dos candidatos (neste caso, todos os vértices) à localização do primeiro equipamento. O menor valor corresponde ao vértice v_9, com custo igual a 19088. Assim, após a primeira iteração, $\mathbf{X}^1=\{v_9\}$. Para localizar o segundo equipamento, é necessário calcular $w_i.\min\{d(v_i, v_9); d(v_i, v_j)\}$ para cada possível candidato $v_j \in \mathbf{V}\backslash\mathbf{X}^1$ e cada vértice de procura $v_i \in \mathbf{V}\backslash(\mathbf{X}^1 \cup \{v_j\})$. Os resultados destes cálculos são apresentados na tabela A3 (anexo A). Os totais das colunas correspondem aos custos associados aos candidatos à localização do segundo equipamento. Agora, o melhor é incorporar na

solução parcial a localização no vértice v_7 com custo igual a 12580. Como solução parcial então obtemos $\mathbf{X}^2=\{v_9, v_7\}$. Para localizar o terceiro equipamento, é necessário calcular $w_i.\min\{d(v_i, v_7); d(v_i, v_9); d(v_i, v_j)\}$ para cada possível candidato $v_j \in \mathbf{V}\backslash\mathbf{X}^2$ e cada vértice $v_i \in \mathbf{V}\backslash(\mathbf{X}^2 \cup \{v_j\})$, calcular o valor da função gulosa para cada candidato e seleccionar aquele que apresentar o menor custo. Procedendo-se desta forma, obtêm-se os resultados mostrados na tabela 3.1 para a localização dos cinco primeiros equipamentos.

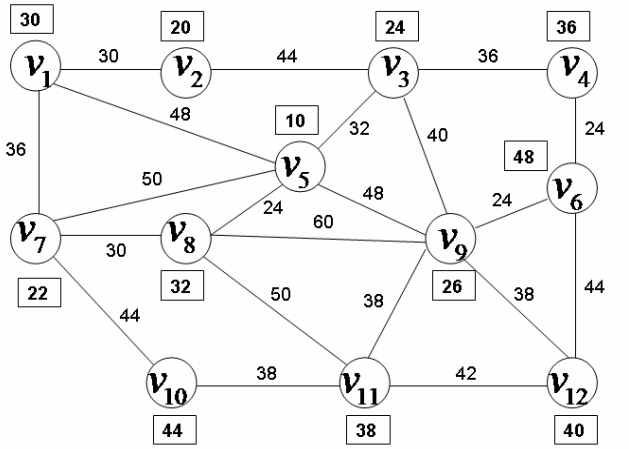

Figura 3.2. Rede para um problema da p-mediana.

Apesar de a solução obtida pelo algoritmo guloso poder não ser a óptima, este algoritmo é apelativo por inúmeras razões. Em primeiro lugar, porque é muito simples de implementar e rápido a executar. Em segundo lugar, porque, na prática, muitas vezes é dada a localização de alguns equipamentos que não podem ser deslocados, sendo o objectivo encontrar a localização de um pequeno número (muitas vezes apenas um ou dois) de novos equipamentos.

Número da localização	Vértice	Custo
1	v_9	19088
2	v_7	12580
3	v_6	10564
4	v_{10}	8628
5	v_1	6828

Tabela 3.1: Resultados para a localização dos primeiros cinco equipamentos na rede da figura 3.2.

3.1.2. Algoritmos Gulosos Aleatorizados

O facto de o algoritmo guloso ser determinístico, isto é, encontrar sempre a mesma solução, se executado diversas vezes para uma dada instância do problema, excepto por eventuais empates, leva a que, no intuito de obter diversidade nas soluções encontradas, se recorra a variantes aleatorizadas deste procedimento. Nesta secção são descritos alguns dos mecanismos de construção que introduzem aleatoriedade ao algoritmo guloso [31, 34].

Algoritmo guloso aleatorizado RPG (random-plus-greedy)

O algoritmo RPG [31] é uma variante do algoritmo guloso que selecciona aleatoriamente $k < p$ (parâmetro de entrada) localizações de equipamentos para a solução e depois completa a solução parcial inicial usando o algoritmo guloso. Na figura 3.3 é apresentado um pseudo-código para este procedimento de construção aplicado ao problema da p-mediana.

```
procedimento RPG (V, p, c(.,.), k) {
    X ← seleccionar aleatoriamente k elementos de V
    enquanto (|X| < p ) {
      calcular c(v_i, X), ∀ v_i ∈ V\X
      v* ← argmin{ c(v_i, X)| v_i ∈ V\X }
      X ← X ∪ {v*}
    }
 retornar X
}
```

Figura 3.3: Pseudo-código do algoritmo guloso aleatorizado RPG para o problema da p-mediana.

Algoritmo guloso aleatorizado com lista restrita de candidatos

Uma das primeiras ideias associadas à aleatorização do algoritmo guloso é a da utilização de uma lista restrita de candidatos (RCL, do termo em inglês Restricted Candidate List) [9] que contém os elementos a que correspondem os melhores valores da função gulosa. Em cada iteração, é escolhido, aleatoriamente, um elemento da lista restrita

para fazer parte da solução parcial. A constituição desta lista pode ser baseada em *cardinalidade* ou em *valor*.

Algoritmo guloso aleatorizado com RCL baseada em cardinalidade: em cada iteração, a lista da qual é escolhido, aleatoriamente, um elemento para fazer parte da solução parcial é constituída por um número fixo de elementos. Na *k*-ésima iteração, em vez de seleccionar o melhor candidato dentre todas as n-k+1 opções, escolhe aleatoriamente de uma lista restrita de candidatos contendo as melhores $\lceil \alpha(n-k+1) \rceil$ opções, onde α, tal que $0 < \alpha \leq 1$, é um parâmetro de entrada. A figura 3.4 mostra um pseudo-código para um procedimento guloso aleatorizado com RCL baseada em cardinalidade.

```
procedimento GulosoAleatorizadoRCL_C (V, p, c(.,.), α ){
    X ← ∅
    para k=1 até p {
      calcular c(v_i, X), ∀ v_i ∈ V\X
      RCL←seleccionar ⌈α(n−k+1)⌉ elementos v_i∈V\X com menor c(v_i,X)
      v ← seleccionar aleatoriamente um elemento da RCL
      X ← X ∪ {v}
     }
 retornar X
}
```

Figura 3.4: Pseudo-código do algoritmo guloso aleatorizado com RCL baseada em cardinalidade para o problema da p-mediana.

Algoritmo guloso aleatorizado com RCL baseada em valor: em cada iteração, a lista é formada pelos elementos candidatos cujos valores da função gulosa pertencem a um dado intervalo. Assim, na *k*-ésima iteração, são determinados o maior e o menor valor da função gulosa para os elementos candidatos, respectivamente c^* e c_*. Uma vez determinados estes valores, é escolhido aleatoriamente, para integrar a solução parcial $\mathbf{X}^k$, um elemento da lista restrita de candidatos (RCL) que contém todos os elementos candidatos v_i cujo valor da função gulosa $c(v_i, \mathbf{X}^{k-1})$ verifica a condição $c(v_i, \mathbf{X}^{k-1}) \leq c_* + \alpha(c^* - c_*)$, onde $0 \leq \alpha \leq 1$ é um parâmetro de entrada. Note-se que se $\alpha = 0$, então este esquema de selecção é um algoritmo guloso, enquanto que se $\alpha = 1$, a selecção do elemento candidato é totalmente aleatório. A figura 3.5 mostra o pseudo-código para um procedimento guloso aleatorizado com RCL baseada no valor.

```
procedimento GulosoAleatorizadoRCL_V (V, p, c(.,.), α ){
    X ← ∅
    para k=1 até p {
      calcular c(v_i, X), ∀ v_i ∈ V\X
      c_* ← min {c(v_i, X) | v_i ∈ V\X }
      c^* ← max {c(v_i, X) | v_i ∈ V\X }
      RCL ← {v_i ∈ V\X| c(v_i, X) ≤ c_* + α(c^* - c_*)}
      v ← seleccionar aleatoriamente um elemento da RCL
      X ← X ∪ {v}
    }
 retornar X
}
```

Figura 3.5: Pseudo-código do algoritmo guloso aleatorizado com RCL baseada em valor para o problema da p-mediana.

Algoritmo guloso aleatorizado com função tendência

Este procedimento (ver [2]) é uma variação da construção gulosa aleatorizada com RCL baseada em valor que, em vez de seleccionar aleatoriamente o elemento candidato da RCL, selecciona-o de acordo com uma distribuição de probabilidade que favorece os elementos com menores custos. Com este fim os elementos candidatos são ordenados numa lista de acordo com os respectivos valores da função gulosa e é utilizada uma *função tendência*, `bias`(.), que associa ao r-ésimo elemento, v_i, desta lista o valor `bias`($r(v_i)$). Uma vez avaliados todos os candidato da RCL, a probabilidade $\pi(r(v_i))$ de o elemento v_i ser seleccionado é

$$\pi\left(r\left(v_i\right)\right)=\frac{bias\left(r\left(v_i\right)\right)}{\sum_{v_j\in RCL} bias\left(r\left(v_j\right)\right)},$$

onde $r(v_i)$ é a posição do elemento v_i na RCL.

Diversas alternativas têm sido propostas para definir a função tendência. Por exemplo:

i) tendência aleatória (`bias` *(r)*=1);

ii) tendência linear (`bias` *(r)*= $\frac{1}{r}$);

iii) tendência logarítmica (`bias` (*r*) = 1/log(*r*+1))

iv) tendência exponencial (`bias` *(r)*=e^{-r}) .

Note-se que todos os métodos de construção gulosos aleatorizados descritos nas secções anteriores utilizam uma função tendência aleatória.

A figura 3.6 mostra um pseudo-código para um procedimento de construção guloso aleatorizado com função tendência aplicado ao problema da p-mediana.

```
procedimento GulosoFuncaoTendencia (V, p, c(., .), α, bias(r(.)) ) {
    X ← ∅
    para k=1 até p {
      calcular c(v_i, X), ∀ v_i ∈ V\X
      c_* ← min {c(v_i, X) | v_i ∈ V\X }
      c^* ← max {c(v_i, X) | v_i ∈ V\X }
      RCL ← {v_i ∈ V\X | c(v_i, X) ≤ c_* + α(c^* - c_*)}
      atribuir posições r(v_i), ∀ v_i ∈ RCL
      v ← seleccionar aleatoriamente um elemento da RCL com
          distribuição de probabilidade π(r(v_i)) = bias(r(v_i)) / Σ_{v_j ∈ RCL} bias(r(v_j))
      X ← X ∪ {v}
    }
 retornar X
}
```

Figura 3.6: Pseudo-código do algoritmo guloso aleatorizado com função tendência para o problema da p-mediana.

Algoritmo guloso aleatorizado puro

O algoritmo guloso aleatorizado puro (*sample greedy*) [34] é semelhante ao algoritmo guloso, mas, em cada iteração, em vez de seleccionar o melhor candidato dentre todas as opções possíveis, considera apenas $q < n$ possíveis inserções, escolhidas aleatoriamente do conjunto de candidatos, seleccionando para ser incorporado na solução aquele que possuir menor custo. A ideia é fazer q suficientemente pequeno para reduzir o tempo de execução em comparação com o algoritmo guloso, mantendo-se um certo grau de aleatorização. Na figura 3.7 é apresentado o pseudo-código desta variante aleatorizada do algoritmo guloso.

```
procedimento GulosoAleatorizadoPuro(V, p, c(., .), q) {
    X ← ∅
    enquanto |X| < p {
      Q← seleccionar aleatoriamente q elementos de V\X
      calcular c(v_i, X), ∀ v_i ∈ Q
      v* ← argmin{ c(v_i, X)| v_i ∈ Q}
      X ← X ∪ {v*}
    }
 retornar X
}
```

Figura 3.7: Pseudo-código do algoritmo guloso aleatorizado puro para o problema da p-mediana.

3.2. Heurísticas de Pesquisa Local

Os algoritmos de pesquisa local são considerados *algoritmos de melhoramento* (ou refinamento) e constituem uma família de técnicas baseadas na noção de vizinhança.

Seja **S** o espaço de pesquisa (conjunto das soluções admissíveis) de um problema de optimização e $c(.)$ a função objectivo a minimizar. A função $\mathcal{N}(.)$, que depende da estrutura do problema tratado, associa a cada solução $\mathbf{X}\in\mathbf{S}$, a sua vizinhança $\mathcal{N}(\mathbf{X})\subseteq\mathbf{S}$. Cada solução $\mathbf{X}'\in\mathcal{N}(\mathbf{X})$ é chamada de *vizinha* de **X**. Denomina-se *movimento* a modificação m que transforma uma solução **X** noutra, **X′**, que esteja na sua vizinhança. Esta operação de modificação da solução **X** representa-se por $\mathbf{X}'\leftarrow\mathbf{X}\oplus m$.

Partindo de uma solução inicial $\mathbf{X}^0$ (obtida, geralmente, por um método construtivo), um algoritmo de pesquisa local procura, na k-ésima iteração, melhorar a solução corrente $\mathbf{X}^{k-1}$ através da exploração de soluções que pertencem a uma determinada *estrutura de vizinhança* de $\mathbf{X}^{k-1}$. O procedimento termina quando não é encontrada na vizinhança uma solução melhor do que a solução corrente. As soluções obtidas deste modo dizem-se *mínimos locais,* no sentido em que minimizam a função objectivo na vizinhança definida, mas não há garantia de que essa solução seja também um *mínimo global* para a função objectivo. A chave do sucesso de um algoritmo de pesquisa local consiste na escolha adequada da estrutura de vizinhança, de técnicas eficientes de pesquisa na vizinhança, e da solução inicial.

3.2.1 Algoritmo de Pesquisa em Vizinhança

O algoritmo de pesquisa em vizinhança (*neighborhood search*) para o problema da *p*-mediana foi proposto pela primeira vez em 1964, por Maranzana [26]. Este algoritmo inicia com qualquer conjunto de *p* localizações de equipamentos. Pode iniciar, por exemplo, com *p* localizações identificadas por um algoritmo construtivo. Conhecido um conjunto de *p* localizações de equipamentos, $\mathbf{V}_p$, independentemente de as localizações serem óptimas ou não, cada vértice de procura v_i é afecto ao equipamento mais próximo v_j, uma vez que o objectivo é minimizar o custo total. Criam-se, desta forma, conjuntos de vértices $\mathbf{P}_j=\{v_i \in \mathbf{V} |\ d(v_i,\ v_j) \leq d(v_i,\ v_k),\ \forall\, v_k\ \in \mathbf{V}_p \backslash \{v_j\}\}$ tais que todos os vértices v_i pertencentes ao mesmo conjunto $\mathbf{P}_j$ estão afectos ao mesmo equipamento v_j. Os vértices de procura pertencentes a um mesmo conjunto $\mathbf{P}_j$ constituem a vizinhança do equipamento v_j a que estão afectos. Para cada localização de equipamento, o algoritmo identifica o conjunto de vértices de procura que constitui a sua vizinhança. De seguida, no interior de cada vizinhança, é encontrada a localização óptima de um equipamento. Se alguma localização de equipamento mudar, o algoritmo reafecta os vértices de procura ao equipamento mais próximo e identifica novas vizinhanças. Se é alterada a constituição de alguma das vizinhanças, o algoritmo, mais uma vez, encontra a localização óptima de um equipamento no interior de cada vizinhança. Este procedimento é repetido até que não ocorreram alterações nas vizinhanças.

A figura 3.8 apresenta o pseudo-código do algoritmo de pesquisa em vizinhança para o problema da p-mediana, onde $\mathbf{X}^0$ é a solução inicial.

```
Procedimento PesquisaVizinhanca(V, c(.), X^0){
   k ← 0
   repetir{
       para cada v_i ∈ X^k {
          N(v_i) ← determinar vizinhança
          resolver 1-mediana em N(v_i)
       }
       X^(k+1) ← relocalizar equipamentos
    }enquanto (X^(k+1) for diferente de X^k)
   // se verifiquem alterações nas localizações dos equipamentos
 Retornar X^k;
}
```

Figura 3.8: Pseudo-código do algoritmo pesquisa em vizinhança para o problema da p-mediana.

A figura 3.9 mostra as vizinhanças associadas às cinco localizações de equipamentos obtidas pelo algoritmo guloso para rede da figura 3.2. A única vizinhança em que a localização de um equipamento não é óptima é a associada ao vértice v_7, pelo que o equipamento é deslocado do vértice v_7 para o vértice v_8. Esta alteração provoca uma redução do custo total de 6828 para 6528. De seguida, é feita uma nova afectação dos vértices de procura ao novo conjunto de equipamentos e verifica-se que o vértice v_5 deve agora ser afecto ao equipamento localizado no vértice v_8. Desta forma, o custo total é reduzido para 6288. As vizinhanças resultantes são apresentadas na figura 3.10.

No interior de cada uma das vizinhanças, procura-se novamente a localização óptima de um equipamento, mas, agora, novas localizações de um equipamento nas respectivas vizinhanças não conduzem a um melhoramento da solução, pelo que as localizações não sofrem alteração e o algoritmo termina.

Uma das limitações associadas ao algoritmo de pesquisa em vizinhança é que, na avaliação do impacto da decisão de qualquer relocalização, apenas são considerados os efeitos nos vértices que pertencem à vizinhança. Os potenciais benefícios para vértices que se encontram fora da vizinhança não são considerados quando se decide se a relocalização deve ser feita ou não.

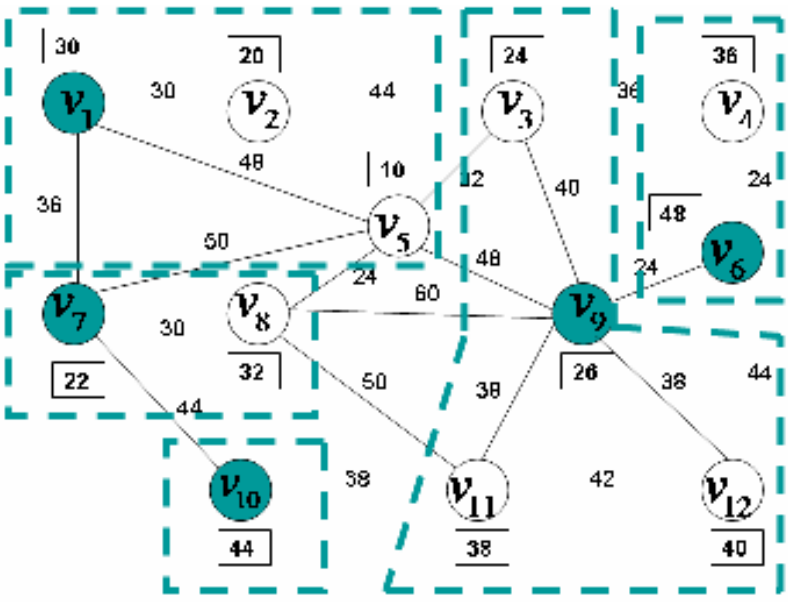

Figura 3.9: Vizinhanças associadas à solução obtida pelo algoritmo guloso para o problema 5-mediana na rede da figura 3.2.

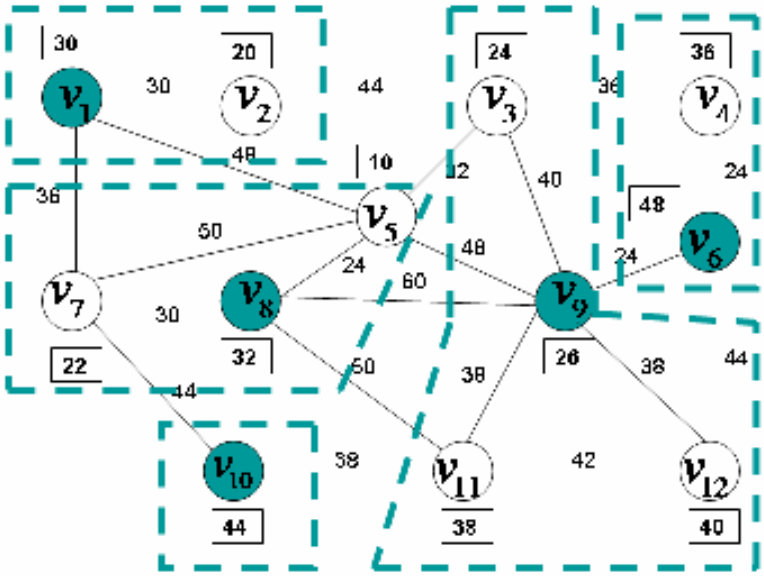

Figura 3.10: Novas vizinhanças resultantes da relocalização do equipamento e da reafectação dos vértices de procura.

3.2.2. Algoritmo de Pesquisa Baseada na Substituição de Vértices

O algoritmo de pesquisa baseada na substituição de vértices (*vertex substitution*) para o problema da p-mediana foi proposto pela primeira vez por Teitz e Bart [37]. Este método parte de uma solução inicial $\mathbf{X}^0$ composta por p localizações de equipamentos e, para cada vértice candidato v_i, encontra, se existir, a localização de equipamento que substituída por v_i melhora ao máximo o valor da função objectivo. Se essa localização de equipamento existe, então os vértices são trocados, e o processo é repetido para a nova solução corrente. Quando todos os vértices candidatos tiverem sido analisados, o algoritmo termina com uma solução que é um mínimo local.

Neste método, a vizinhança da solução corrente é definida como o conjunto de soluções admissíveis alcançadas pela troca de uma localização presente na solução por outra que não faz parte dela. As estratégias mais usadas na substituição da solução corrente são *melhor aperfeiçoamento*, em que todos os vizinhos são analisados e o melhor é escolhido como óptimo local, e *primeiro aperfeiçoamento*, o algoritmo move-se para uma solução vizinha logo que uma primeira melhoria seja encontrada. Se uma solução melhor é encontrada, repete-se o processo, considerando esta nova solução.

Na figura 3.11 é apresentada uma descrição do algoritmo de pesquisa baseada na substituição de vértices.

```
Procedimento SubstituicaoVertices(V, c(.), X^0){
   X_p ← X^0
   X* ← X^0
   repetir {
       para cada v_i ∈ V\X_p{
            para cada v_j ∈ X_p{
                 X ← (X_p\{v_j}) ∪ {v_i}
                 calcular c(X)
            }
            v_sub ← argmin{ c((X_p\{v_j}) ∪ {v_i}) | v_j ∈ X_p}
            se c((X_p\{v_sub}) ∪ {v_i}) < c(X*){
                 X_p ← (X_p\{v_sub}) ∪ {v_i}
                 X* ← (X_p\{v_sub}) ∪ {v_i}
            }
       }
   } enquanto (existirem vértices não analisados)
   Retornar X*
}
```

Figura 3.11: Descrição do algoritmo pesquisa baseada na substituição de vértices.

Para ilustrar este método considere-se a solução obtida pelo algoritmo de pesquisa em vizinhança para o problema da 5-mediana na rede da figura 3.2. A tabela 3.2 apresenta, para cada um dos sete vértices que não estão presentes na solução, a melhor das cinco localizações de equipamentos a ser substituída. A solução apresentada na figura 3.10 envolve a localização dos equipamentos nos vértices v_1, v_6, v_8, v_9 e v_{10} com um custo total de 6288. Se for considerada a inserção do vértice v_2, a melhor localização de equipamento a remover para ser substituída por v_2 é a localização v_1. Esta troca aumenta o custo total em 300 unidades, ou seja, para 6588. As entradas para a solução quer do vértice v_5 quer do vértice v_7 também degradam o valor da função objectivo. No entanto, as inserções dos vértices v_3, v_4, v_{11} ou v_{12}, em substituição da localização de equipamento v_9, melhoram o valor da função objectivo em 96, 96, 276 e 512 unidades, respectivamente. Destas, a troca que origina o menor valor da função objectivo é a da localização de equipamento v_9 pelo vértice v_{12}. Esta mudança é efectuada e é obtida uma nova solução com os equipamentos localizadas nos vértices v_1, v_6, v_8, v_{10} e v_{12}, com custo total de 5776. Esta solução é mostrada na figura 3.12.

Vértice a inserir	Equipamento a ser substituída	Custo total	Diferença*
v_2	v_1	6588	300
v_3	v_9	6192	- 96
v_4	v_9	6192	- 96
v_5	v_9	6720	432
v_7	v_8	6828	540
v_{11}	v_9	6012	- 276
v_{12}	v_9	5776	- 512

Tabela 3.2: Sumário das oportunidades de troca para a solução obtida pelo algoritmo pesquisa em vizinhança para a rede da figura 3.2.

* Valores positivos indicam uma degradação no valor da função objectivo; valores negativos indicam uma melhoria.

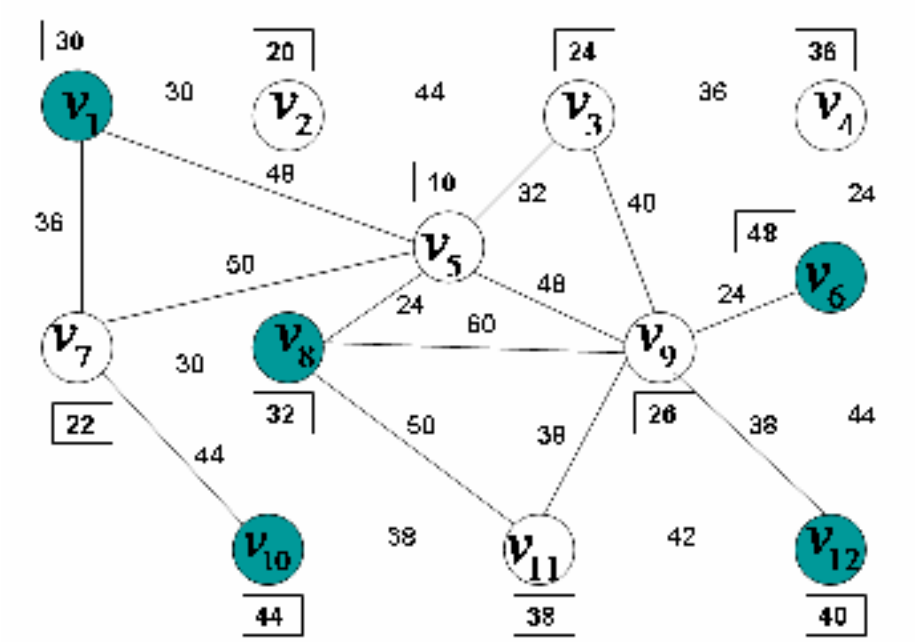

Figura 3.12: Solução obtida depois da aplicação do algoritmo de pesquisa baseada na substituição de vértices.

3.3. Metaheurísticas

Durante as duas últimas décadas, desenvolveram-se distintos métodos heurísticos para resolução de problemas de optimização combinatória, alguns dos quais são estruturas genéricas para a exploração do espaço de soluções admissíveis do problema que tratam de optimizar. Este tipo de heurística é usualmente conhecido por *metaheurística* e a sua utilização na resolução de problemas de optimização visa suprir limitações das heurísticas tradicionais, como a paragem prematura em óptimos locais, por exemplo. Procura-se, assim, uma maior aproximação da solução obtida ao óptimo global. Por isso, na procura da melhor solução, ou seja, da melhor aproximação ao óptimo global, é possível que uma metaheurística abandone temporariamente um óptimo local, mesmo que isso motive uma perda temporária na qualidade da solução corrente.

Nesta secção serão abordados, de forma resumida, os princípios básicos de alguns dos procedimentos metaheurísticos mais utilizados e reconhecidos em optimização combinatória: pesquisa tabu, pesquisa em vizinhança variável, algoritmos genéticos e GRASP (procedimento de pesquisa gulosa aleatorizado e adaptativo).

3.3.1. Pesquisa Tabu

A metaheurística pesquisa tabu é uma técnica utilizada para resolver problemas combinatórios de grande dificuldade baseada em princípios gerais de Inteligência Artificial. A sua origem situa-se em diversos trabalhos publicados há cerca de 20 anos. As ideias básicas da pesquisa tabu foram sugeridas por Hansen, em 1986 [19] mas, oficialmente, o nome e a metodologia foram introduzidos por Fred Glover, em 1989 [13].

Pesquisa tabu [38] é uma técnica que pode ser utilizada para guiar qualquer procedimento de pesquisa local na procura agressiva da solução óptima, procurando evitar a paragem prematura num óptimo local. Para tal, este método toma da Inteligência Artificial o conceito de memória e implementa-o mediante estruturas simples com o objectivo de dirigir a pesquisa tendo em conta a sua história. Isto é, o procedimento extrai informações do sucedido em iterações anteriores e actua em conformidade. Neste sentido, pode-se dizer que existe uma certa aprendizagem e que a pesquisa é inteligente.

O esquema geral da pesquisa tabu é o seguinte: A partir de uma solução admissível inicial, determina-se uma solução vizinha que seja melhor do que ela até obter-se uma solução cuja qualidade seja considerada aceitável. Nesta metaheurística, mais do que falar de vizinhos de uma solução é conveniente falar do conjunto de movimentos que permite construir uma solução a partir de outra.

Dada uma solução $\mathbf{X} \in \mathbf{S}$, define-se um *movimento* como um elemento $m \in \mathbf{M}(\mathbf{X})$ que permite construir uma solução vizinha: $\mathbf{X}' = \mathbf{X} \oplus m \in \mathcal{N}(\mathbf{X}) \ \ \forall m \in \mathbf{M}(\mathbf{X})$.

Começando com uma solução inicial admissível $\mathbf{X}^0$, um algoritmo de pesquisa tabu explora, em cada iteração, um subconjunto **C** da vizinhança $\mathcal{N}(\mathbf{X})$ da solução corrente **X**. Para um problema de minimização, a solução $\mathbf{X}' \in \mathbf{C}$**,** vizinha de **X,** para a qual a função objectivo $c(.)$ toma o menor valor em **C**, torna-se a nova solução corrente, mesmo que $\mathbf{X}'$ seja pior do que **X** (isto é, mesmo que $c(\mathbf{X}') > c(\mathbf{X})$). Assim, uma das características principais desta metaheurística é a aceitação de movimentos que levem a soluções piores, para evitar a paragem prematura em óptimos locais. Por outro lado, o processo de substituição da solução corrente pelo melhor vizinho pode originar ciclos, isto é, o retorno do algoritmo a uma solução que já foi visitada anteriormente. Para evitar que isto aconteça, é definida uma lista **T** de movimentos proibidos, chamada *lista tabu*. Geralmente, a lista tabu contém os movimentos contrários aos últimos $|\mathbf{T}|$ (parâmetro de entrada do método) movimentos realizados. A lista tabu é uma fila de tamanho fixo: *quando um novo movimento é adicionado à lista, o mais antigo sai*. Assim, na exploração do subconjunto **C** da vizinhança $\mathcal{N}(\mathbf{X})$ da solução corrente **X**, ficam logo excluídos da pesquisa os vizinhos $\mathbf{X}'$ que são obtidos de **X** por movimentos que constam na lista tabu.

Se por um lado a lista tabu reduz o risco de surgimento de ciclos (uma vez que garante o não-retorno a uma solução visitada anteriormente durante $|\mathbf{T}|$ iterações), por outro, também pode não permitir movimentos para soluções de boa qualidade que ainda não foram visitadas [37]. Por isso, existe um mecanismo chamado *função de aspiração* que retira, sob certas condições, o estatuto de proibido a um movimento. Mais precisamente, para cada possível valor c_{valor} da função objectivo $c(.)$ existe um nível de aspiração $A(c_{valor})$ tal que uma solução $\mathbf{X}' \in \mathbf{C}$ pode ser gerada se $c(\mathbf{X}') \leq A(c(\mathbf{X}))$, mesmo que o movimento m pelo qual $\mathbf{X}'$ é obtido a partir de **X** esteja na lista tabu. A actualização da função de

aspiração permite flexibilizar a pesquisa, variando o nível de proibição à medida que o algoritmo evolui.

Geralmente são usados dois critérios de paragem: 1) parar quando é atingido um certo número máximo de iterações sem melhora no valor da função objectivo; 2) parar quando o valor da função objectivo da melhor solução encontrada atinge um limite inferior conhecido (ou se aproxima dele). Este segundo critério evita a execução desnecessária do algoritmo quando uma solução óptima é encontrada ou quando é obtida uma solução considerada suficientemente boa.

A figura 3.13 apresenta o pseudo-código do algoritmo de pesquisa tabu geral para um problema de optimização que minimiza uma dada função objectivo $c(.)$.Os principais parâmetros de entrada do algoritmo pesquisa tabu são:

i) uma solução inicial admissível, $\mathbf{X}^0$;

ii) o número de movimentos proibidos da lista tabu, $|\mathbf{T}|$;

iii) a função de aspiração, $A(.)$;

iv) o número de soluções vizinhas da solução corrente que serão testadas em cada iteração, $|\mathbf{C}|$;

v) o número máximo de iterações sem melhora no valor da função objectivo, PT_{Max};

vi) um valor mínimo para a função objectivo, c_{min} para ser usado no segundo critério de paragem.

```
procedimento PesquisaTabu ( X^0 , |T|, A(.), |C|, c_min, PT_Max ){
    X ← X^0 , X* ← X
    k ← 0 , k* ← 0
    T ← ∅
    inicializar a função de aspiração A(.)
    enquanto ( k − k* < PT_max ∧ c(X) > c_min ){
      k ← k + 1
      C ← gerar |C| soluções X'=X⊕m de N(X): m ∉T ∨ c(X')< A(c(X))
      calcular c(X'), ∀ X'∈ C
      X' ← argmin{c(X')|X'∈ C}
      T ← T\{ m_mais-antigo}∪{ m_mais-recente}
      actualizar a função de aspiração A(.)
      X ← X'
      se (c(X)< c(X*))
        X* ← X, k* ← k
    }
    Retornar X*
}
```

Figura 3.13: Pseudo-código do algoritmo pesquisa tabu geral para um problema de minimização

Diversas variantes da metaheurística pesquisa tabu foram propostas para resolver o problema da p-mediana [28, 36]. Em [36], Rolland, Schilling e Current apresentam uma proposta, na qual são permitidos dois tipos de movimentos: *movimentos de adição* (*ADD*) e *movimentos de perda* (*DROP*). Dada uma solução **X**, são considerados dois conjuntos: o conjunto das localizações de equipamentos seleccionadas, **X**, e o conjunto das potenciais localizações de equipamentos não seleccionadas, **V****X**. Um movimento consiste na selecção e transferência de um vértice de um dos conjuntos para o outro. Se o vértice é movido para **X**, o movimento é designado *movimento de adição*, se o vértice é movido de **X**, o movimento é chamado *movimento de perda*. É definido *movimento de troca* (*SWAP*) como a permuta de dois vértices, um de **X** e outro de **V****X** (essencialmente um par movimento de adição / movimento de perda). Assim, são considerados dois tipos de vizinhança da solução corrente **X**: *vizinhanças construtivas* que resultam da adição de um vértice de **V****X** ao conjunto das localizações de equipamentos **X**; e *vizinhanças destrutivas* que resultam da perda de uma localização de equipamento por parte de **X**.

Em geral, na pesquisa tabu, quando é efectuado um movimentos permitido, é considerado tabu o movimento que conduz à soluções já investigada, no entanto, nesta proposta, estas restrições estão associadas apenas ao movimento de adição. Isto é, uma vez adicionado ao conjunto das localizações dos equipamentos, um vértice é classificado como tabu.

Relativamente à função de aspiração, é usada a que determina que se um movimento produz uma solução melhor do que a melhor solução gerada (e a solução resultante é admissível), então é-lhe retirado o estatuto de tabu e o movimento é executado.

3.3.2. Pesquisa em Vizinhança Variável (PVV)

A metaheurística pesquisa em vizinhança variável (*Variable Neighborhood Search,* VNS), proposta em [20] e [21], consiste na exploração do espaço de soluções admissíveis através de mudanças sistemáticas das estruturas de vizinhança. Contrariamente ao que sucede com outras metaheurísticas baseadas em métodos de pesquisa local, as quais exploram soluções numa vizinhança fixa de uma solução corrente, o método PVV explora estruturas de vizinhanças variáveis construídas com base numa métrica que define a distância entre duas soluções do espaço de pesquisa.

Um algoritmo de pesquisa em vizinhança variável para o problema da p-mediana foi proposto por Mladenovic e Hansen em [18]. Neste algoritmo as k_{max} estruturas de vizinhança, $k_{max} \leq p$, são construídas com base numa *função de distância simétrica,* ρ, definida no espaço de soluções admissíveis **S**. Dadas duas soluções admissíveis quaisquer $\mathbf{X}'$, $\mathbf{X}'' \in \mathbf{S}$, cada uma delas composta por p localizações de equipamentos, define-se distância entre $\mathbf{X}'$ e $\mathbf{X}''$, representada por $\rho(\mathbf{X}', \mathbf{X}'')$, como o número de localizações em que estas duas soluções diferem, isto é,

$$\rho(\mathbf{X}', \mathbf{X}'') = |\mathbf{X}' \setminus \mathbf{X}''| = |\mathbf{X}'' \setminus \mathbf{X}'|, \quad \forall \quad \mathbf{X}', \mathbf{X}'' \in \mathbf{S} \tag{3.2}$$

Assim, para $k = 1, \ldots, k_{max}$, $\mathbf{X}' \in \mathcal{N}_k(\mathbf{X})$, se e somente se, $\rho(\mathbf{X}, \mathbf{X}') = k$, ou seja, se $\mathbf{X}$ e $\mathbf{X}'$ diferem exactamente em k equipamentos. A ideia básica da metaheurística PVV é então a exploração incremental, para $k = 1, \ldots, k_{max}$, do conjunto de estruturas de vizinhanças da solução admissível $\mathbf{X}$, $\mathcal{N}(\mathbf{X}) = \{\mathcal{N}_1(\mathbf{X}), \ldots, \mathcal{N}_{kmax}(\mathbf{X})\}$.

A figura 3.14 apresenta o pseudo-código do algoritmo geral da metaheurística pesquisa em vizinhança variável, que pode ser aplicado no problema da p-mediana. O algoritmo parte de uma solução inicial admissível $\mathbf{X}^0$ e, na k-ésima iteração, selecciona aleatoriamente (de forma a evitar ciclos) um vizinho $\mathbf{X}'$ pertencente à vizinhança $\mathcal{N}_k(\mathbf{X})$ da solução corrente **X**. Esse vizinho é então submetido a um procedimento de pesquisa local. Se a solução óptima local, $\mathbf{X}''$, for melhor que a solução corrente **X**, a pesquisa continua a partir da exploração das estruturas de vizinhança de $\mathbf{X}''$, $\mathcal{N}(\mathbf{X}'')$, recomeçando com a primeira estrutura de vizinhança $\mathcal{N}_1(\mathbf{X}'')$. Caso contrário, a pesquisa continua a partir da estrutura de vizinhança seguinte da solução corrente **X**, $\mathcal{N}_{k+1}(\mathbf{X})$. O procedimento é interrompido quando um critério de paragem for atingido. Os critérios mais utilizados neste método são três e dependem de:

i) tempo máximo de processamento;

ii) número máximo de iterações;

iii) número máximo de iterações consecutivas entre dois melhoramentos.

```
procedimento PVV ( X^0 , k_max ){
    X ← X^0
    enquanto (critério de paragem não for satisfeito){
      k ← 1
      enquanto ( k ≤ k_max ) {
        X′ ← seleccionar aleatoriamente uma solução vizinha em N_k(X)
        X″ ← pesquisaLocal( X′ )
        se c(X″) < c(X) {
           X ← X″
           k ← 1
        }
        senão k ← k+1
      }
    }
 retornar X
}
```

Figura 3.14: Pseudo-código do algoritmo geral de pesquisa em vizinhança variável.

3.3.3. Algoritmos Genéticos (AG)

Os algoritmos genéticos [6, 8, 16, 29] são métodos aproximados de resolução de problemas de optimização combinatória, baseados na teoria da selecção natural e evolução das espécies, que utilizam conceitos da adaptação natural e da reprodução selectiva dos organismos. Introduzidos por Holland, em 1975, no seu trabalho "Adaptation in Natural and Artificial Systems" [22], os algoritmos genéticos usam a analogia entre a resolução de problemas e os processos naturais de evolução. Dada uma população, os indivíduos com melhores características genéticas têm mais hipóteses de sobrevivência e de produzirem descendentes cada vez mais aptos, enquanto indivíduos menos aptos (com piores características genéticas) tendem a desaparecer.

Num algoritmo genético cada *cromossoma* (indivíduo da população) contém a codificação de uma possível solução do problema. A forma como as soluções são codificadas varia em função das características do problema em questão, sendo, em geral, usada a codificação binária. Nos algoritmos genéticos com codificação binária, os indivíduos são representados por vectores de dígitos binários, onde cada elemento de um vector denota a presença (1) ou ausência (0) de um determinado atributo. Cada um desses atributos é conhecido como *gene*. Os possíveis valores que um determinado gene pode

assumir são denominados *alelos*. No caso particular do problema da p-mediana, cada cromossoma é, geralmente, representado por uma cadeia constituída pelos *p* índices dos vértices seleccionados para localizações de equipamentos. Considere-se, por exemplo, uma rede com 15 vértices (potenciais localizações de equipamentos) representados pelos índices 1, 2, …, 15, onde se pretende localizar cinco equipamentos. O cromossoma [2, 5, 7, 10, 15] representa a solução admissível, em que os vértices com índices 2, 5, 7, 10 e 15 são seleccionados para localizações dos equipamentos.

Ao contrário dos métodos anteriormente descritos, os quais, em cada iteração, operam com uma única solução, um algoritmo genético opera com uma população de soluções admissíveis (cromossomas). O algoritmo parte de uma população inicial de indivíduos, $\mathbf{P}^0$, gerada aleatoriamente, que evolui através de iterações sucessivas, chamadas *gerações*, até encontrar a solução óptima ou uma solução suficientemente próxima da óptima. Na *k*-ésima iteração é gerada uma nova população $\mathbf{P}^k$ através de uma fase de *reprodução* de alguns indivíduos da população corrente $\mathbf{P}^{k-1}$. Esta fase de reprodução consiste na selecção de indivíduos da geração corrente $\mathbf{P}^{k-1}$ para operações de *cruzamento* e/ou *mutação*. A selecção de indivíduos pode ser aleatória ou baseada no valor de uma certa *função aptidão* que avalia a qualidade de cada indivíduo enquanto solução do problema. A selecção de bons cromossomas para progenitores orienta o algoritmo para as melhores regiões do espaço de pesquisa, uma vez que a informação (material genético) por eles transportada será transferida para os seus descendentes.

Operação Cruzamento

A operação *cruzamento* tenta simular o processo de cruzamento de genes, cuja finalidade é combinar adequadamente cromossomas de progenitores aptos à reprodução, de modo a criar descendentes. Baseado neste processo evolutivo, num algoritmo genético os cromossomas de dois progenitores (cromossomas pais) são combinados de forma a gerar cromossomas filhos (normalmente dois) através da cópia de partes dos seus genes. Na sua forma mais básica, o cruzamento é feito através da identificação aleatória de um ponto de cruzamento nos cromossomas pais e da troca das secções dos dois cromossomas pais situadas depois desse ponto, o que resulta na geração de dois filhos. A figura 3.15 mostra a operação de cruzamento dos cromossomas pais $\mathbf{X}_{p1}$ e $\mathbf{X}_{p2}$ da qual resultam os cromossomas filhos $\mathbf{X}_{f1}$ e $\mathbf{X}_{f2}$. Neste caso, o ponto de cruzamento é três. Os três componentes iniciais do

pai $\mathbf{X}_{p1}$ foram mantidos no filho $\mathbf{X}_{f1}$ cuja cadeia é complementada com os dois componentes finais do pai $\mathbf{X}_{p2}$. A cadeia do filho $\mathbf{X}_{f2}$ é composta pelos três primeiros componentes do pai $\mathbf{X}_{p2}$ mais os dois últimos do pai $\mathbf{X}_{p1}$.

$\mathbf{X}_{p1} = [2, 5, 7, 10, 15]$	$\Rightarrow$	$\mathbf{X}_{f1} = [2, 5, 7, 12, 14]$
$\mathbf{X}_{p2} = [1, 5, 6, 12, 14]$		$\mathbf{X}_{f2} = [1, 5, 6, 10, 15]$

Figura 3.15: Operação cruzamento de dois progenitores.

Esta maneira particular de transferir genes de pais para filhos é um exemplo do operador cruzamento geralmente referido como *operador cruzamento básico* ou *de um ponto*. Existem outros operadores cruzamento, nomeadamente operador cruzamento *de dois pontos* e cruzamento *uniforme*.

Operação Mutação

Os filhos resultantes da operação cruzamento podem ser sujeitos a uma operação mutação que consiste na alteração aleatória de uma parte dos genes de cada cromossoma (ou componentes da solução). O operador mutação tem como objectivo introduzir variedade genética na população e, portanto, evitar que o algoritmo pare prematuramente num óptimo local.

A figura 3.16 apresenta um exemplo de mutação bastante simples, no qual o terceiro componente do cromossoma sofre uma modificação, o seu valor passou de 7 para 9.

$$[2, 5, 7, 12, 14] \Rightarrow [2, 5, 9, 12, 14]$$

Figura 3.16: Operação mutação

Ambas as operações genéticas, *cruzamento* e *mutação*, são realizadas com uma certa probabilidade, sendo a probabilidade de ocorrência da primeira muito mais elevada

do que a da segunda, que, geralmente, ronda 1%. Gerados os descendentes da população $\mathbf{P}^k$, define-se a população sobrevivente, isto é, as $|\mathbf{P}^0|$ soluções que integrarão a população da k+1-ésima geração.

Selecção da população sobrevivente

Para determinar a população sobrevivente, é utilizada uma *função aptidão* que quantifica a aptidão dos indivíduos que constituem a população de soluções candidatas para a resolução do problema considerado. Quando o problema a resolver é um problema da p-mediana, a função de aptidão é, em geral, a função objectivo. A aptidão de um indivíduo é dada pelo valor da função objectivo para a solução representada por esse indivíduo. Normalmente, na escolha dos indivíduos sobreviventes são usados os seguintes critérios: *i*) *aleatório*; *ii*) *roleta*, onde a hipótese de sobrevivência de um cromossoma é proporcional ao seu nível de aptidão, e *iii*) *misto*, isto é, uma combinação dos dois critérios anteriores. Em qualquer um destes critérios é admitida a sobrevivência de indivíduos menos aptos, numa tentativa de escapar a óptimos locais.

O método de selecção mais utilizado é o da roleta. Por analogia com o processo evolucionista de Darwin, o método tende a escolher para sobreviver à geração seguinte as soluções mais aptas. Na natureza, todos os indivíduos têm hipóteses de sobreviver à geração seguinte. No entanto, os mais dominantes, com maior aptidão, têm maior probabilidade de sobreviver, dando origem aos novos potenciais progenitores da próxima geração, enquanto que os que estão menos aptos se extinguem.

Critério de Paragem

Todo o procedimento descrito anteriormente pode ser repetido geração atrás de geração até que um certo critério de paragem seja satisfeito. O método termina quando um dos seguintes critérios se verifica: 1) um número máximo de gerações é atingido; 2) a melhor solução encontrada atinge um certo nível de aptidão; 3) não há melhoria da solução durante um certo número de iterações.

A figura 3.17 apresenta o pseudo-código de um algoritmo genético básico. Uma observação imediata é a existência de um número significativo de parâmetros de entrada: *i*) o tamanho da população inicial, *ii*) a função aptidão; *iii*) as probabilidades de ocorrência

dos operadores de cruzamento e mutação; *iv*) o tipo de operador de cruzamento; *v*) o método de selecção de progenitores; *vi*) o critério de paragem.

```
procedimento AlgoritmoGenético( ){
    k←0
    P^0 ← gerar população inicial
    Avaliar aptidões dos indivíduos de P^0
    enquanto (critério de paragem não for satisfeito){
        k← k+1
        P^k ← gerar população a partir de P^(k-1)
        Avaliar aptidões dos indivíduos de P^k
        P^k ← seleccionar população sobrevivente
    }
    X ← seleccionar a melhor solução de P^k
    retornar X
}
```

Figura 3.17:Pseudo-código de um algoritmo genético básico

3.3.4. Procedimento Guloso Aleatorizado e Adaptativo (GRASP)

O procedimento guloso aleatorizado e adaptativo, conhecido por GRASP (abreviado de *Greedy Randomized Adaptative Search Procedure*), foi proposto por Feo e Resende em [10]. GRASP é outra metaheurística usada para encontrar boas aproximações da solução óptima de um problema de optimização combinatória e tem sido muito aplicada na resolução do problema da p-mediana. A metaheurística GRASP é um método multi-arranque. Isto significa que em cada iteração obtém uma solução admissível diferente. É um método desenhado em duas fases: uma *fase de construção*, na qual uma solução admissível é gerada, elemento a elemento, e uma *fase de melhoramento*, na qual se recorre a um algoritmo de pesquisa local para procurar um óptimo local na vizinhança da solução construída. Este procedimento repete-se um determinado número de vezes (parâmetro de entrada do método) e a melhor solução encontrada no conjunto de todas as iterações GRASP é devolvida como solução aproximada do problema.

Fase de Construção

No caso particular do problema da p-mediana, na fase de construção do GRASP é normalmente utilizado o algoritmo guloso aleatorizado com lista restrita de candidatos (RCL) baseada em cardinalidade, descrito na subsecção 3.1.2. Na utilização deste procedimento de construção reside a componente probabilística do GRASP, uma vez que o elemento seleccionado para integrar a solução é escolhido aleatoriamente dentre os melhores candidatos da RCL, pelo que este elemento pode não ser o melhor candidato existente. Assim, repetidas aplicações do procedimento de construção originam diferentes soluções iniciais para a pesquisa local.

Fase de Melhoramento

Dado que a solução gerada pelo procedimento de construção GRASP pode não ser um óptimo local com respeito à definição de vizinhança adoptada, é benéfico aplicar uma pesquisa local para tentar melhorar a solução construída. Assim, na fase de melhoramento do GRASP pode ser aplicado qualquer procedimento de pesquisa local, nomeadamente o procedimento de pesquisa local baseado na substituição de vértices, proposto por Teitz e Bart e descrito na subsecção 3.2.2. No entanto, note-se que a metaheurística GRASP se baseia na realização de múltiplas iterações e na manutenção da melhor solução encontrada, pelo que não é especialmente vantajoso que esta se detenha demasiado tempo no melhoramento de uma dada solução.

Uma característica particularmente apelativa do GRASP é a facilidade com que ele pode ser implementado. São poucos os parâmetros que precisam de ser definidos e ajustados e, por isso, o algoritmo pode focar-se na execução eficiente das estruturas de dados, de modo a assegurar iterações GRASP rápidas. O pseudo-código apresentado na figura 3.18 ilustra o procedimento GRASP para um problema de minimização em geral. Os parâmetros de entrada são apenas dois:

i) o número máximo de iterações GRASP, *maxItr*, usado como critério de paragem,

ii) o parâmetro α que determina o número de elementos da LRC, no procedimento guloso aleatorizado com RCL baseada em cardinalidade.

```
procedimento GRASP (maxItr, α){
    c* ←+∞
    para k=1 até maxItr {
      X ←ConstrucaoGulosaAleatorizadaComRCL_C()
      X ←PesquisaBaseadaSubstituicaoVertices(X)
      se ( c(X)< c*){
        c* ← c(X)
        X*←X
      }
    }
    retornar X*
}
```

Figura 3.18: Pseudo-código GRASP básico.

4. Um Algoritmo Híbrido para o Problema da p-Mediana

Neste capítulo é descrito um algoritmo híbrido para o problema da p-mediana proposto por Resende e Werneck em [34].

4.1. Descrição Geral

Chamado pelos seus autores de híbrido, este método, como o seu nome indica, combina elementos de diversas metaheurísticas tradicionais com o objectivo de encontrar soluções próximas do óptimo para o problema da p-mediana. Na sua essência este algoritmo é similar ao GRASP, ou seja, é um método iterativo multi-arranque, onde em cada iteração é usado um algoritmo construtivo guloso aleatorizado para gerar uma solução admissível, que é, de seguida, submetida a um processo de melhoramento por pesquisa local. Como é habitual num algoritmo multi-arranque, a melhor solução obtida em todas as iterações é apresentada como resultado final. Esta abordagem básica é reforçada no algoritmo híbrido com algumas estratégias de intensificação:

1. Com o objectivo de concentrar a pesquisa em determinadas regiões consideradas promissoras é mantido um grupo com algumas das melhores soluções encontradas nas iterações já executadas, as chamadas *soluções elite*.
2. Em cada iteração, a solução obtida por pesquisa local é combinada com uma solução elite através de um processo chamado *religação por caminhos* [14, 15, 25].
3. Uma vez concluídas as iterações da fase multi-arranque existe uma segunda fase, a *fase de pós-optimização*, em que as soluções elite são *combinadas* umas com as outras.

A figura 4.1 resume o algoritmo híbrido. Como já foi referido, este método combina elementos de várias metaheurísticas. À semelhança do GRASP, é uma abordagem multi-arranque em que cada iteração consiste basicamente num procedimento construtivo guloso

aleatorizado seguido de um procedimento de melhoramento por pesquisa local [9, 10, 32]. À metaheurística pesquisa tabu, o método foi buscar a ideia de *religação por caminhos* [15, 25]. Além disso, é usado no melhoramento da religação por caminhos o conceito de *múltiplas gerações*, uma característica-chave dos algoritmos genéticos [16, 27].

Os parâmetros de entrada deste método são: *i*) o número máximo de iterações GRASP, *maxItr*, usado como critério de paragem, e *ii*) o número de soluções elite, |GSE|.

```
procedimento Hibrido (maxItr, |GSE| ){
    GSE ← ∅          // inicializar lista com soluções elites
    para k=1 até maxItr {
      X ← construirAleatorizado( )
      X ← pesquisaLocal(X)
      X′ ← seleccionar(GSE, X)  // o elemento a ser combinado com X
      se ( X′ ≠ NULL ) {
        X′ ← religarCaminhos(X, X′)
        inserir(GSE, X′)
       }
      inserir(GSE, X)
    }
    X ← Pós-optimização(GSE)
    retornar X
}
```

Figura 4.1: Pseudo-código do algoritmo híbrido.

De seguida, é feita a análise de como cada uma das três componentes deste método, *algoritmo construtivo*, *pesquisa local* e *intensificação*, pode ser implementada para resolver o problema da p-mediana.

4.2. Algoritmo Construtivo Aleatorizado

Nos testes levados a cabo pelos autores deste algoritmo, em diferentes instâncias do problema da p-mediana, para escolherem o melhor método a usar, na fase de construção do híbrido, foram consideradas diferentes variantes aleatorizadas do algoritmo guloso. Dentre os métodos mais rápidos, o algoritmo guloso aleatorizado puro (*sample greedy*), proposto por Resende e Werneck em [34] e descrito na subsecção 3.1.2, foi aquele que apresentou soluções com uma qualidade ligeiramente superior. O valor do parâmetro *q* (número de

possíveis inserções na solução parcial, escolhidas aleatoriamente do conjunto de candidatos) considerado foi $q = \lceil \log_2 (n/p) \rceil$, onde n representa o número de elementos candidatos e p o número de equipamentos a localizar.

Pelo facto de, no método híbrido, qualquer investimento extra na construção da solução poder tornar o algoritmo mais lento, sem ganhos significativos na qualidade das soluções encontradas, a estratégia guloso aleatorizado com RCL baseada em cardinalidade, normalmente usada no GRASP, não foi escolhida. Embora esta construa soluções com melhor qualidade que o método guloso aleatorizado puro, é mais lenta do que este.

4.3. Pesquisa Local

Para procedimento de pesquisa local, é adoptada uma implementação do método de pesquisa baseada na substituição de vértices descrito na subsecção 3.2.2, proposta recentemente por Resende e Werneck [33]. Esta técnica de melhoramento utiliza a estratégia *melhor aperfeiçoamento*: todos os vizinhos são analisados e o melhor é escolhido para substituir a solução corrente. Apesar de ter a mesma complexidade, no pior caso, que a implementação proposta por Whitaker [39], na qual é usada a estratégia *primeiro aperfeiçoamento* (o algoritmo interrompe a exploração da vizinhança quando o primeiro vizinho melhor é encontrado), a implementação adoptada no algoritmo híbrido pode ser substancialmente mais rápida na prática. A aceleração resulta do uso de informação obtida nas iterações iniciais do algoritmo, a qual permite reduzir o custo computacional em etapas posteriores.

4.4. Intensificação

Durante a execução do algoritmo híbrido, as soluções encontradas cuja qualidade seja considerada boa formam o grupo de soluções elite (GSE).

Conforme mostra o pseudo-código do algoritmo (figura 4.1), o processo de intensificação ocorre em duas fases:

1. em cada iteração da fase multi-arranque a recém gerada solução é combinada com uma solução seleccionada do grupo de soluções elite;

2. na fase de pós-optimização, as soluções elite são combinadas entre si para se obter a solução resultante.

Em ambas as fases, a estratégia utilizada para combinar um par de soluções admissíveis é a mesma: uma técnica conhecida na literatura especializada como *religação por caminhos*. Esta técnica originalmente foi proposta por Glover [15] como uma forma de explorar trajectórias entre soluções elite obtidas por pesquisa tabu. Posteriormente este procedimento foi aplicado no contexto de uma heurística GRASP por Laguna e Martii [25], e, desde então, tem sido amplamente implementado em diferentes metaheurísticas (em [32] são apresentados numerosos exemplos).

Seguidamente proceder-se-á à descrição da técnica religação por caminhos.

4.4.1. Religação por Caminhos

A combinação de soluções através de *religação por caminhos* é feita do seguinte modo. Considere-se $\mathbf{X}'$ e $\mathbf{X}''$ duas soluções admissíveis, interpretadas como conjuntos de localizações de equipamentos. O procedimento começa com uma das soluções (digamos, $\mathbf{X}'$) e gradualmente transforma-a na outra ($\mathbf{X}''$) através de *movimentos de troca* [36]. Com este fim, são seleccionados para fazer parte de $\mathbf{X}'$ equipamentos que pertencem a $\mathbf{X}''$, mas que não fazem parte de $\mathbf{X}'$ (elementos de $\mathbf{X}''\setminus \mathbf{X}'$) e são trocados por equipamentos que pertencem a $\mathbf{X}'$ mas que não fazem parte de $\mathbf{X}''$ (elementos de $\mathbf{X}'\setminus \mathbf{X}''$). O número total de trocas feitas, $|\mathbf{X}''\setminus \mathbf{X}'| = |\mathbf{X}'\setminus \mathbf{X}''|$, é a *distância simétrica* entre $\mathbf{X}'$ e $\mathbf{X}''$. A escolha da troca a fazer em cada fase é gulosa, isto é, realiza-se sempre o movimento mais rentável (ou menos dispendioso).

O resultado da *religação por caminhos* é o melhor mínimo local encontrado no caminho de $\mathbf{X}'$ para $\mathbf{X}''$. Neste contexto, é considerado um *mínimo local* uma solução tal que, quer a solução que lhe sucede no caminho quer a que a antecede são estritamente piores do que ela. Se o caminho não tem mínimo local, uma das soluções originais ($\mathbf{X}'$ ou $\mathbf{X}''$) é apresentada como resultado, com igual probabilidade. Quando há uma melhoria da solução no caminho, este critério corresponde exactamente ao critério usual e então é apresentado como resultado o melhor elemento no caminho. A diferença existe apenas

quando a *religação por caminhos* usual não tem sucesso. Neste caso tenta-se aumentar a diversidade das soluções exploradas através da selecção de uma solução que não seja um dos extremos do caminho.

Um aspecto importante da *religação por caminhos* é a *direcção* em que esta é efectuada. Dadas as soluções $\mathbf{X}'$ e $\mathbf{X}''$ deve-se decidir se se executa a *religação por caminhos* de $\mathbf{X}'$ para $\mathbf{X}''$ ou de $\mathbf{X}''$ para $\mathbf{X}'$. Neste algoritmo, a *religação por caminhos* é executado da melhor para a pior solução dentre as duas, na fase multi-arranque, e da pior para a melhor solução, na fase de pós-optimização.

Note-se que a *religação por caminhos* é muito semelhante ao procedimento de pesquisa local baseada na substituição de vértices proposto por Teitz e Bart em [37], com duas diferenças. Primeiro, o número de movimentos autorizados é restrito: apenas elementos em $\mathbf{X}''\backslash\mathbf{X}'$ podem ser inseridos, e apenas elementos em $\mathbf{X}'\backslash\mathbf{X}''$ podem ser removidos. Segundo, são permitidos movimentos que não melhoram a solução corrente.

O procedimento de intensificação é, ainda, aumentado com a realização de uma pesquisa local completa na solução produzida por *religação por caminhos*. A aplicação de pesquisa local, neste ponto do algoritmo, aumenta a diversidade, uma vez que podem ser utilizados equipamentos que não pertencem a nenhuma das soluções originais. Note-se que este procedimento tem algumas semelhanças com a metaheurística pesquisa em vizinhança variável, descrita na subsecção 3.3.2. Partindo de um óptimo local, a pesquisa em vizinhança variável obtém uma solução em algumas vizinhanças alargadas e aplica pesquisa local tentando encontrar uma solução melhor. A principal diferença é que a pesquisa em vizinhança variável selecciona aleatoriamente a solução vizinha, enquanto que aqui é usado como guia um segundo óptimo local. Neste caso, a distância entre a nova solução e a original (na verdade para ambos os extremos) é pelo menos dois.

4.4.2. Gestão do Grupo de Soluções Elite

Um aspecto importante do algoritmo é a gestão do grupo de soluções elite (GSE). Empiricamente, observa-se que a aplicação da religação por caminhos a um par de soluções tem menor probabilidade de sucesso se as soluções forem muito similares, isto é, se a distância simétrica entre elas for muito pequena [34]. Quanto maior o caminho

percorrido entre duas soluções, maior a probabilidade de ser encontrado um mínimo local completamente diferente das soluções originais. É, portanto, razoável que na gestão do GSE se tenha em conta não só a qualidade das soluções, mas também a sua diversidade.

A gestão do grupo de soluções elite é feita através de duas operações essenciais:

1. ***Inserção:*** Para que uma solução **X** com custo $c(\mathbf{X})$ possa ser adicionada ao GSE, duas condições devem ser satisfeitas:
   - *i.* A distância simétrica de **X** com cada uma das soluções de elite **X′,** cujo custo seja inferior a $c(\mathbf{X})$ tem de ser, pelo menos quatro, ou seja:
     $$\rho(\mathbf{X}, \mathbf{X}') = |\mathbf{X} \setminus \mathbf{X}'| = |\mathbf{X}' \setminus \mathbf{X}| \geq 4, \ \forall \ \mathbf{X}' \in \mathbf{GSE}: \ c(\mathbf{X}') \leq c(\mathbf{X})$$
   - *ii.* Se o GSE está completo, a solução **X** tem de ser pelo menos tão boa como a pior solução elite (se o GSE não está completo, isso não é, obviamente, necessário).

   Assim, para que uma solução **X** com custo $c(\mathbf{X})$ possa ser adicionada ao GSE, primeiramente deve ser analisado se o GSE está completo ou não. Se o GSE está completo e ambas as condições, *i* e *ii*, são satisfeitas, a solução **X** é inserida no GSE, substituindo a solução mais semelhante dentre as que possuem custo igual ou superior a $c(\mathbf{X})$. Se o GSE não está completo e a nova solução **X** não dista menos de quatro de qualquer outra solução elite, **X** é simplesmente acrescentada ao GSE.

2. ***Selecção:*** Em cada iteração do algoritmo, é seleccionada uma solução do GSE para ser combinada com **X**, a solução mais recentemente encontrada. Uma estratégia de selecção aplicada com algum sucesso a outros problemas é a selecção uniformemente aleatória da solução. No entanto, a adopção desta estratégia de selecção, possibilitaria a selecção, para combinação por religação por caminhos com **X**, de soluções elite muito similares a **X**, sendo improvável, nestes casos, encontrar uma boa solução. No sentido de minimizar este problema, uma solução **X′** é seleccionada do GSE com probabilidade proporcional à sua distância simétrica, $\rho(\mathbf{X}, \mathbf{X}')$, relativamente à solução **X**.

4.4.3. Pós-optimização

O processo de pós-optimização é realizado ao concluir a fase multi-arranque. Nessa fase, o processo de procura de uma boa solução não produz um, mas vários diferentes óptimos locais, que, frequentemente, não são muito piores que a melhor solução encontrada. Todas estas soluções fazem parte do grupo de soluções elite, cuja construção foi descrita anteriormente. O objectivo da fase de pós-optimização é combinar as soluções do GSE na tentativa de obter outras ainda melhores. Assim, esta fase tem como entrada o grupo de soluções elite, GSE, e cada solução $\mathbf{X}' \in$GSE é combinada com cada uma das outras soluções elite, $\mathbf{X}'' \in$GSE, usando o método de religação por caminhos. Como resultado, as soluções geradas por este processo formam um novo grupo de soluções elite que representam uma *nova geração* de soluções. O algoritmo prossegue iterativamente até ser criada uma geração de soluções elite que não melhora as gerações anteriores.

5. O Problema da Gestão Óptima da Diversidade

Neste capítulo será feita uma descrição do problema da gestão óptima da diversidade, ODMP (do inglês Optimal Diversity Management Problem), que pode ser visto como um caso particular do problema da p-mediana.

5.1. Exemplo de Aplicação Industrial

O problema da gestão óptima da diversidade tem origem na indústria de cablagens eléctricas para automóveis. Os construtores de automóveis oferecem aos seus clientes um grande número de opções (*airbag* condutor e/ou passageiro, *airbag* lateral, volante à direita ou à esquerda, ar condicionado, auto-rádio, sensores de estacionamento, etc. …), e permitem-lhes que as combinem como entenderem, dentro dos limites do exequível. Por exemplo, um automóvel não pode possuir simultaneamente volante à esquerda e à direita, ou ser equipado com *airbag* para o passageiro se não possui *airbag* para o condutor. Os automóveis são equipados com as ligações necessárias para activar o conjunto de opções solicitadas pelo cliente. A cada opção corresponde uma ligação física (fio de cobre). Da liberdade de escolha dada aos clientes resulta um número elevado de configurações possíveis, entendendo configuração como um conjunto de ligações. De facto, este número pode chegar a vários milhares como mostra a tabela 5.1. Igualmente elevado é o número de configurações que ocorrem na prática. Por razões técnicas, está fora de questão lidar com tantas configurações de cablagem diferentes na linha de montagem. Para além disso, a produção de todas as configurações seria certamente muito dispendiosa. Por isso, na prática, das diferentes configurações consideradas, produz-se apenas um número muito menor que, dependendo do fabricante, pode variar entre 6 e 100 configurações. Caso seja necessária uma configuração não produzida, ela é substituída pela configuração compatível (que permite o funcionamento de todas as opções para as quais a cablagem substituída foi concebida) de menor custo dentre as produzidas. Nestes casos, são fornecidos aos clientes automóveis equipados com configurações que normalmente possuem mais ligações do que

as necessárias para activar as opções solicitadas. Esta operação de substituição implica que alguns fios da cablagem substituta não sejam utilizados, e por consequência um custo adicional de produção (custo associado às ligações desnecessárias).

.

Número de opções	Número aproximado de configurações
10	1 000
20	1 000 000
30	1 000 000 000
40	1 000 000 000 000

Tabela 5.1: Relação entre o número de opções e o número de configurações, ignorando restrições técnicas.

5.2. Enunciado e Notações

Seja **V** um conjunto cujos elementos serão designados *configurações*, em referência ao exemplo de aplicação industrial apresentado na secção anterior, onde cada configuração i representa o conjunto das opções que i activa. Considere-se definida em **V** uma relação de ordem parcial $\prec$ tal que $i \prec j$ se e somente se a configuração i pode ser substituída pela configuração j e $i \neq j$. Diz-se que $i \preceq j$ se $i \prec j$ ou $i = j$.

Para cada configuração $i \in \mathbf{V}$, seja $c_i > 0$ o custo unitário de produção de i e $d_i \geq 0$ o número de exemplares de i que devem ser produzidos.

Dadas duas configurações i e j, verifica-se sempre a proposição $i \prec j \Rightarrow c_i < c_j$.

O custo resultante da substituição da configuração i pela configuração $j \succ i$ é igual a $d_i c_j$, e o custo adicional é igual a $d_i(c_j - c_i)$.

Por fim, seja p o número de configurações diferentes que se pretende produzir.

O problema da gestão óptima da diversidade (ODMP) consiste em determinar quais as p configurações que devem ser produzidas e quais as substituições a operar de forma a minimizar o custo adicional total resultante dessas substituições. Note-se que minimizar o custo adicional total ou o custo total é equivalente, pois estes dois valores diferem numa constante. Com efeito, no óptimo, a diferença entre o custo total e o custo adicional total é

igual ao custo resultante da produção de cada configuração separadamente, ou seja, ao custo da diversidade maximal onde $p = |\mathbf{V}|$.

Este problema é NP-difícil (ver [1] e [4]).

5.3. Grafo Associado

Considere-se o grafo orientado $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ associado à relação de ordem parcial $\prec$. Os elementos do conjunto de vértices $\mathbf{V}$ representam as configurações. O custo w_{ij} associado a cada arco (i, j) de $\mathbf{A} = \{(i, j) \in \mathbf{V}\times\mathbf{V} | \ i \prec j\}$ é dado pelo produto da procura da configuração i , d_i, pelo custo de produção da configuração j, c_j, isto é, $w_{ij} = c_j d_i$ [3].

Neste trabalho, serão utilizados indiferentemente os termos vértices ou configurações, assim como arcos ou substituições.

A figura 5.1 mostra o grafo orientado associado ao ODMP no caso em que cada configuração é um conjunto de ligações capazes de activar três opções possíveis. Cada configuração é representada por um vector com três componentes que só podem tomar os valores 0 e 1, onde 1 significa que a configuração permite o funcionamento da opção associada.

A algumas configurações pode estar associado um pedido nulo. Contudo a sua produção pode ser útil, uma vez que podem substituir outras configurações pedidas. Representa-se por $\mathbf{V}^*$ o conjunto das configurações que possuem pedidos não nulos.

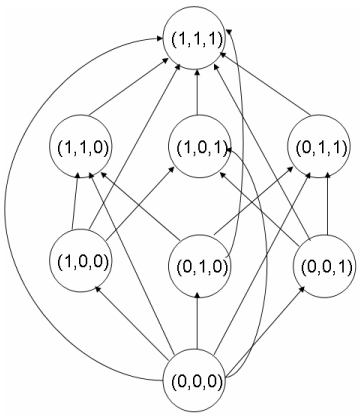

Figura 5.1: Grafo orientado associado ao ODMP com três opções possíveis.

As configurações que não podem ser substituídas por nenhuma outra configuração dizem-se *folhas*. São os elementos maximais relativamente à ordem parcial. Os elementos minimais denominam-se *raízes*. No exemplo da figura 5.1, a configuração *(0,0, 0)* é a raiz, e *(1, 1, 1)* é a única folha.

5.4. Modelação

O ODMP é um caso particular de diversos problemas difíceis bem conhecidos. Por exemplo, pode ser visto como um problema da p-mediana num grafo com a relação de ordem parcial $\prec$ apresentada anteriormente.

A abordagem feita neste trabalho baseia-se no modelo de programação linear inteira para o problema da p-mediana.

Para $i \prec j$, fazer $x_{ij} = 1$ se a configuração i for substituída pela configuração j e $x_{ij} = 0$ caso contrário. Quando $i = j$, $x_{ii} = 1$ significa que a configuração i é produzida.

Por uma questão de simplificação da notação, considerar-se-á a minimização do custo total da produção.

O problema pode ser modelado pela seguinte programação linear inteira:

minimizar $$c = \sum_{j \in V} \sum_{i \preceq j} c_j d_i x_{ij}$$

sujeito a :

$$\sum_{j \succeq i} x_{ij} = 1 \ , \forall \ i \in \mathbf{V}^* \qquad (5.1)$$

$$\sum_{i \in V} x_{ii} = p \qquad (5.2)$$

$$x_{ij} \leq x_{jj} \ , \ \forall \ i, j \in \mathbf{V}, \ i \prec j \qquad (5.3)$$

$$x_{ij} \geq 0 \ , \ \forall \ i, j \in \mathbf{V}, \ i \prec j \qquad (5.4)$$

$$x_{jj} \in \{0, 1\} \ , \forall \ j \in \mathbf{V} \qquad (5.5)$$

A equação (5.1) assegura que uma configuração que possua pedido não nulo ou é produzida ou é substituída por exactamente uma outra configuração. A equação (5.2) declara que devem ser produzidas exactamente p configurações. A inequação (5.3) assegura que, para que seja possível substituir a configuração i pela configuração j, a configuração j terá de ser produzida. As últimas condições impõem a integralidade das variáveis, que só é

necessária para as variáveis x_{jj}. Se os x_{jj} forem inteiros, então, em qualquer solução extrema do programa linear resultante, as restantes variáveis também o serão.

6. Estudo Computacional

Neste capítulo, efectua-se um estudo computacional de modo a avaliar:

i) o desempenho de diferentes versões do algoritmo híbrido proposto por Resende e Werneck em [34] e do algoritmo guloso, implementado em [1], em termos de qualidade das soluções obtidas e dos tempos de processamento;

ii) o balanço entre a qualidade das soluções obtidas e o custo computacional (tempos de processamento), para cada uma das versões do algoritmo híbrido, tomando como referência os resultados apresentados pelo algoritmo guloso para as instâncias do problema da gestão óptima da diversidade testadas.

Simultaneamente, tentar-se-á averiguar até que ponto algumas características das instâncias testadas, como o número de vértices do grafo ou a razão percentual entre o número de configurações a produzir (p) e o número de vértices do grafo (n), influenciam o desempenho de cada um dos algoritmos.

Relativamente à heurística híbrido, são consideradas três variantes, obtidas através da atribuição de diferentes valores a dois parâmetros que alteram significativamente o seu comportamento: *número de iterações multi-arranque* e *número de soluções elite*. Refira-se que qualquer diminuição nos valores destes parâmetros faz com que o método se torne mais rápido, e que qualquer aumento faz com que as soluções obtidas sejam melhores.

As variantes do híbrido testadas são:

- Híbrido com 1 iteração da fase multi-arranque e 0 soluções elite, denominada H_1Itr_0El. Esta versão do híbrido consiste na construção aleatorizada de uma solução seguida de pesquisa local. Uma vez que o número de soluções elite é 0, não é executada a religação por caminhos.
- Híbrido com 5 iterações da fase multi-arranque e 3 soluções elite, denominada H_5Itr_3El.
- Híbrido com 32 iterações da fase multi-arranque e 10 soluções elite, denominada H_32Itr_10El.

Em primeiro lugar, será feita uma comparação entre os resultados obtidos, para cada instância, pelas heurísticas guloso e híbrido, e os resultados obtidos pelo software de optimização Xpress Optimizer. Ao mesmo tempo, tentar-se-á compreender até que ponto o número de vértices do grafo e a razão percentual entre o número de configurações a produzir (p) e o número de vértices do grafo (n) influenciam o desempenho de cada um dos algoritmos. Por último, cada uma das versões do algoritmo híbrido é comparada com o algoritmo guloso em termos de balanço entre a qualidade das soluções obtidas e os tempos de processamento.

6.1. Ambiente de Teste e Instâncias Utilizadas

Nos testes computacionais foi utilizada uma implementação em C do algoritmo guloso proposto por Agra e al. em [1]. Quanto ao algoritmo híbrido, foi utilizada a implementação da autoria de Resende e Werneck, disponibilizada no sítio da Internet http://www.research.att.com/~mgcr/popstar/popstar.html. Ambas as heurísticas foram executadas numa máquina Intel(R)Core(TM)2Duo, CPU T7300, 2.00GHz com 1,0 GB de RAM.

Na obtenção do óptimo de cada instância, utilizou-se o software Xpress Optimizer 19.00.00, executado na mesma máquina.

Para a comparação das heurísticas foram utilizadas instâncias cujos grafos orientados foram gerados aleatoriamente por um gerador de grafos com os seguintes parâmetros de entrada:

- *Número de opções* (*op*), parâmetro que determina o número de vértices (configurações) do grafo.

- *Percentagem* (*perc*), parâmetro considerado na geração dos vértices do grafo, sendo que, para cada configuração *i*, caso o valor gerado aleatoriamente pelo processador seja inferior ao valor de *perc*, a configuração é gerada, não sendo gerado caso contrário.

- *Mínimo da procura* (*minproc*) e *máximo da procura* (*maxproc*), valores entre os quais são geradas as procuras de cada uma das configurações.

- *Mínimo do custo* (*mincusto*) e *máximo do custo* (*maxcusto*), valores entre os quais são gerados os custos unitários dos fios que permitem activar cada uma das opções.

A tabela 6.1 mostra as combinações de parâmetros com que foram gerados cada um dos 20 grafos utilizados nos testes.

Nome	*op*	*perc*	*minproc*	*maxproc*	*mincusto*	*maxcusto*
Grafo1	6	1	0	15	2	8
Grafo2	6	1	0	150	2	8
Grafo3	6	1	0	15	2	80
Grafo4	6	1	0	150	2	80
Grafo5	7	1	0	15	2	8
Grafo6	7	1	0	150	2	8
Grafo7	7	1	0	15	2	80
Grafo8	7	1	0	150	2	80
Grafo9	8	1	0	15	2	8
Grafo10	8	1	0	150	2	8
Grafo11	8	1	0	15	2	80
Grafo12	8	1	0	150	2	80
Grafo13	9	1	0	15	2	8
Grafo14	9	1	0	150	2	8
Grafo15	9	1	0	15	2	80
Grafo16	9	1	0	150	2	80
Grafo17	10	1	0	15	2	8
Grafo18	10	1	0	150	2	8
Grafo19	10	1	0	15	2	80
Grafo20	10	1	0	150	2	80

Tabela 6.1: Valores dos parâmetros de entrada com que foram gerados os grafos das instâncias testadas

Para cada um dos grafos gerados consideraram-se 7 valores para o número de configurações a produzir (*p*), correspondentes a 5%, 10%, 15%, 20%, 25%, 30% e 35% do número de vértices do grafo.

As instâncias utilizadas nos testes foram organizadas em 5 grupos:

- **Grupo I**, que contém 4 grafos orientados (Grafo1, Grafo2, Grafo3 e Grafo4) com 64 vértices cada e *p* toma os valores 3, 6, 10, 13, 16, 19 e 22.

- **Grupo II**, que contém 4 grafos orientados (Grafo5, Grafo6, Grafo7 e Grafo8) com 128 vértices cada e *p* toma os valores 6,13, 19, 26, 32, 38 e 45.

- **Grupo III**, que contém 4 grafos orientados (Grafo9, Grafo10, Grafo11 e Grafo12) com 256 vértices cada e *p* toma os valores 13, 26, 38, 51, 64, 77 e 90.

- **Grupo IV**, que contém 4 grafos orientados (Grafo13, Grafo14, Grafo15 e Grafo16) com 512 vértices cada e *p* toma os valores 26, 51, 77, 102, 128, 154 e 179.

- **Grupo V**, que contém 4 grafos (Grafo17, Grafo18, Grafo19 e Grafo20) com 1024 vértices cada e p toma os valores 51, 102, 154, 205, 256, 307 e 358.

Por limitações do software, não foi possível desenvolver testes para instâncias de dimensão maior.

Apenas foram consideradas instâncias com grafos conexos, ou seja, grafos em que quaisquer dois dos seus vértices estão ligados por um caminho. Daí que, na geração dos grafos, se tenha atribuído ao parâmetro percentagem, sempre, o valor 1.

6.2. Resultados Computacionais

Nos testes computacionais efectuados, o Xpress apresentou a solução óptima para 121 das 140 instâncias utilizadas. Para as instâncias em que, após 30 minutos de processamento do Xpress, não foi possível encontrar o óptimo, a melhor solução dentre as obtidas pelos algoritmos utilizados nos testes (incluindo o Xpress) será designada *melhor solução obtida*.

As tabelas B1 a B5 (anexo B) apresentam os resultados obtidos para os diversos grupos de instâncias anteriormente descritos. As primeiras quatro colunas indicam, por esta ordem, o nome da instância (*Nome*), o número de vértices do grafo (*n*), o número de configurações a produzir (*p*) e o custo da solução óptima do problema (C_O). Nas instâncias em que não foi possível encontrar a solução óptima, o valor de C_O corresponde ao custo da *melhor solução obtida*, encontrando-se estes valores em itálico. Para cada algoritmo são apresentados três valores: primeiro, o custo da solução obtida (*C*); segundo, a percentagem de desvio do custo da solução obtida pelo algoritmo relativamente ao custo da solução óptima (*d*), sendo $d = 100 \times \frac{C - C_O}{C_O}$; terceiro, o tempo de processamento em segundos (*t*). Note-se que na definição da percentagem de desvio, quando não foi obtida a solução óptima, usa-se o custo da melhor solução admissível obtida e não um limite inferior porque

nem sempre foi possível obter o valor da relaxação linear da instância em causa no limite máximo de tempo.

Qualidade das Soluções

Na avaliação da qualidade das soluções obtidas por cada um dos algoritmos, será utilizada a medida p*ercentagem de desvio* (*d*) definida anteriormente. A média das percentagens de desvio obtida pelo algoritmo para todas as instâncias do grupo considerado designar-se-á *percentagem de desvio média*.

Os resultados apresentados nas tabelas mostram que, como seria de esperar, a variante do híbrido, H_32Itr_10El, é claramente a que apresenta soluções mais próximas das soluções óptimas dos problemas testados, encontrando soluções cujas percentagens de desvio, *d*, não ultrapassam 0,2%. Apresenta, também, uma percentagem de desvio igual a 0 em 136 das 140 instâncias testadas, o que significa que nesses problemas apresenta a melhor solução obtida. Dessas 136 soluções (melhores soluções obtidas), 117 são comprovadamente soluções óptimas. A elevada qualidade das soluções obtidas por este algoritmo é confirmada pela percentagem de desvio média apresentada no conjunto de todas as instâncias testadas, que é inferior a 0,002%.

Em segundo lugar, no que concerne à qualidade das soluções obtidas, surge o algoritmo H_5Itr_3El, que apresenta percentagem de desvio igual a 0 em 86 instâncias. As percentagens de desvio das suas soluções para as instâncias utilizadas nos testes são sempre inferiores a 0,8%. A média das percentagens de desvio apresentadas nas 140 instâncias não ultrapassa os 0,08%. Estes resultados mostram que também este algoritmo obtém soluções muito próximas das soluções óptimas dos problemas testados. O algoritmo H_1Itr_0El surge como o terceiro melhor no que diz respeito à qualidade das soluções encontradas. Com percentagem de desvio igual a 0 em 33 das 140 instâncias utilizadas nos testes, uma percentagem de desvio máxima de 3,61% e uma percentagem de desvio média inferior a 0,6%, este algoritmo, à semelhança dos dois anteriores, apresenta soluções para os problemas relativamente próximas das soluções óptimas. Com maiores desvios percentuais entre os custos das soluções encontradas e os custos das soluções óptimas, aparece, em último lugar, o algoritmo guloso. Embora obtenha a solução óptima em 10 das 140 instâncias, apresenta uma percentagem de desvio máxima (8,84%) e uma percentagem

de desvio média (2,92%) muito superiores às apresentadas pelos demais algoritmos, o que é demonstrativo do seu pior desempenho em termos de qualidade das soluções obtidas.

O gráfico da figura 6.1 mostra, para cada algoritmo, as percentagens de desvio máxima (*d máxima)* e média (*d média*), obtidas a partir dos resultados apresentados pelos algoritmos, para todas as instâncias.

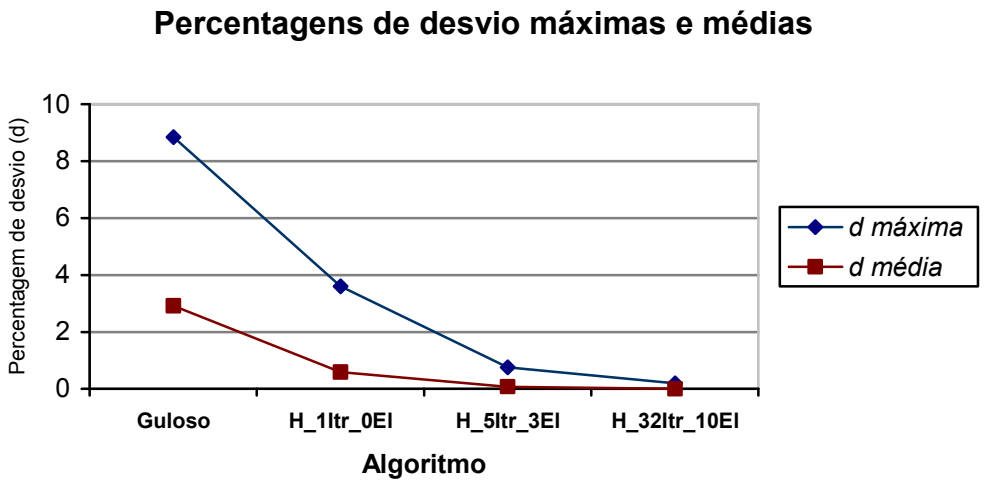

Figura 6.1: Percentagens de desvio máxima e média, por algoritmo.

Com o objectivo de compreender até que ponto o número de vértices do grafo do problema influencia a qualidade das soluções obtidas pelos algoritmos, apresenta-se no gráfico da figura 6.2, para cada grupo de instâncias e cada algoritmo, a percentagem de desvio média (*d*).

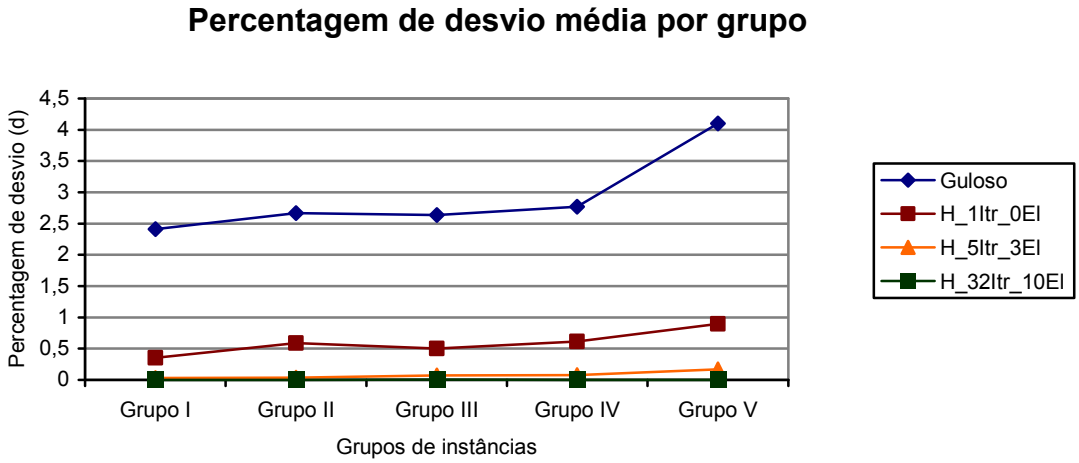

Figura 6.2: Percentagem de desvio média, por algoritmo e grupo de instâncias.

A informação contida no gráfico da figura 6.2 confirma o que já foi dito relativamente às percentagens de desvio das soluções obtidas pelos algoritmos. De facto, em todos os grupos de instâncias, a percentagem de desvio média apresentada pelo H_32Itr_10El é 0 ou muito próxima de 0. Verifica-se, também, que, em todos os grupos de instâncias, as percentagens de desvio médias apresentada pelas variantes do algoritmo híbrido são inferiores a 1%, enquanto que a apresentada pelo algoritmo guloso é superior a 2,4%. É de destacar a elevada percentagem de desvio média obtida pelo algoritmo guloso no Grupo V, 4,1%. Este gráfico mostra, também, que a qualidade das soluções obtidas pelos algoritmos tende a piorar à medida que o número de vértices do grafo aumenta. Em termos absolutos, o aumento do valor da percentagem de desvio média é mais notório para o algoritmo guloso, que passa de 2,4%, no Grupo I (grafos com 64 vértices), para mais de 4%, no Grupo V (grafos com 1024 vértices).

Uma outra característica das instâncias que pode influenciar a qualidade das soluções obtidas pelos algoritmos é a razão percentual entre o número de configurações a produzir (p) e o número de vértices do grafo (n). Como já foi referido, para cada grafo gerado, consideraram-se 7 valores para p, correspondentes a 5%, 10%, 15%, 20%, 25%, 30% e 35% do número de vértices do grafo. O gráfico da figura 6.3 apresenta a relação existente entre o tipo de instância, no que à razão percentual entre p e n diz respeito, e a percentagem de desvio média para cada tipo de instância.

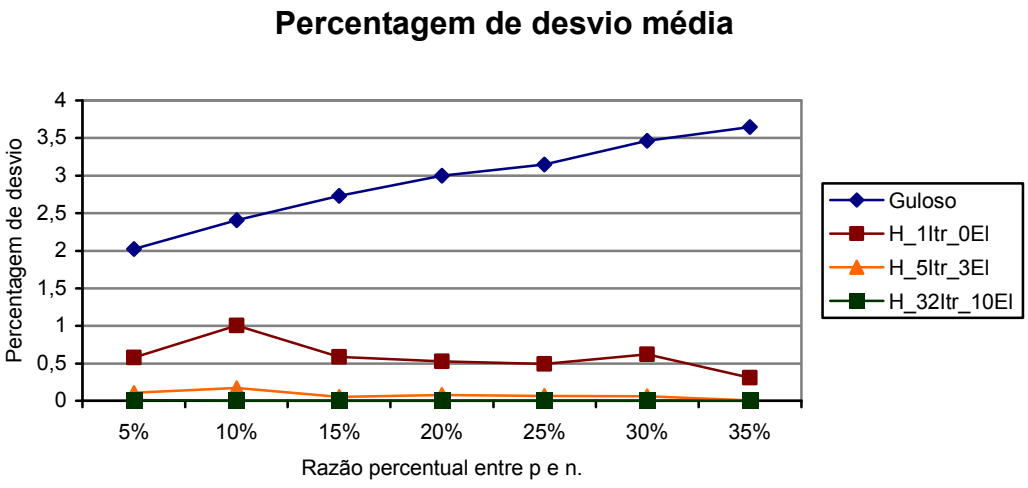

Figura 6.3: Percentagem de desvio média, por algoritmo e tipo de instância.

A partir da observação do gráfico da figura 6.3 conclui-se que, contrariamente ao que acontece nas variantes do algoritmo híbrido, onde o aumento do valor da razão percentual entre *p* e *n* não implica maiores percentagens de desvio médias, parecendo mesmo existir uma ligeira diminuição, no algoritmo guloso é clara a diminuição da qualidade das soluções obtidas à medida que aquela percentagem aumenta. Para este algoritmo, a percentagem de desvio média quase que duplica quando se passa do grupo de instâncias cuja razão percentual entre *p* e *n* é 5% para o grupo de instâncias cuja razão é 35%.

Tempos de Processamento

Na comparação dos tempos de processamento (*t*) obtidos pelos algoritmos, serão utilizados os resultados apresentados nas tabelas B1 a B5 (anexo B).

Observa-se claramente que, como seria de esperar, H_32Itr_10El se destaca por ser mais lento que os demais algoritmos em todas as instâncias testadas. Apresenta um tempo de processamento médio (média dos tempos de processamento do algoritmo na resolução dos 140 problemas testados) de 2 segundos, chegando a atingiu os 12,59 segundos de processamento na resolução de um dos problemas testados. Segue-se, em termos de tempos de processamento, o algoritmo H_5Itr_3El com 1,09 segundos de tempo de processamento máximo e uma média de 0,26 segundos de processamento na resolução dos problemas utilizados nos testes. Das variantes do híbrido, H_1Itr_0El surge como aquela que apresenta os menores tempos de processamento, não demorando mais de 0,31 segundos a resolver qualquer das instâncias testadas. O seu tempo de processamento médio é de 0,09 segundos. Quanto ao guloso, é nitidamente o mais rápido uma vez que apresenta, em todas as instâncias, o menor tempo de processamento. O seu tempo de processamento médio é de apenas 0,03 segundos e o tempo de processamento máximo 0,22 segundos.

O gráfico da figura 6.4 mostra, para cada algoritmo, os tempos de processamento máximo e médio.

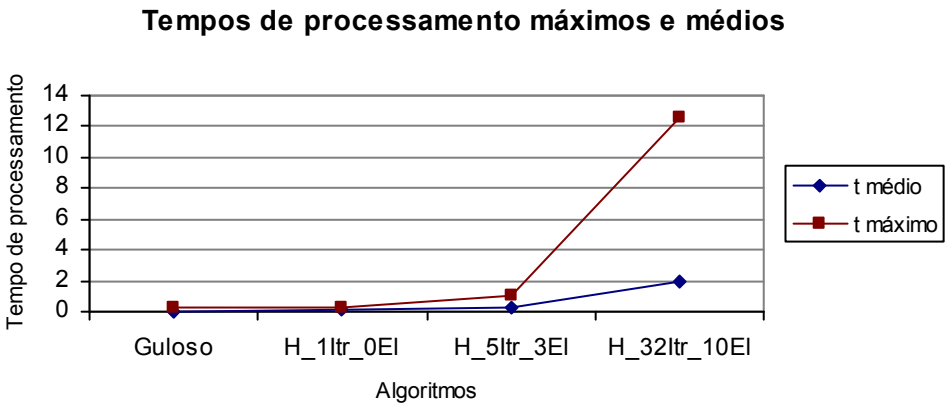

Figura 6.4: Tempos de processamento máximo e médio, por algoritmo.

À semelhança do que foi feito aquando da avaliação da qualidade das soluções obtidas pelos algoritmos, apresenta-se, no gráfico da figura 6.5, para cada algoritmo, o tempo de processamento médio em cada grupo de instâncias.

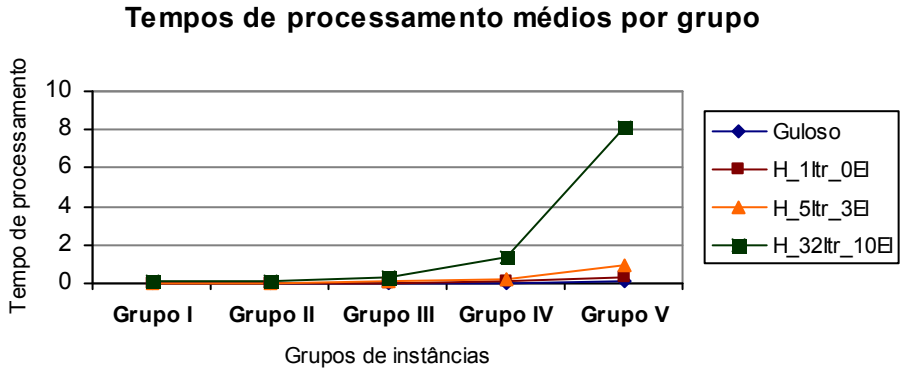

Figura 6.5: Tempo de processamento médio, por algoritmo e grupo de instâncias.

Observa-se que, como seria de esperar, à medida que o número de vértices do grafo aumenta, também o tempo de processamento dos algoritmos aumenta. No entanto, é visível que o aumento do número de vértices do grafo do problema a resolver provoca um maior incremento no tempo de processamento do algoritmo H_32Itr_10El do que nos dos restantes algoritmos. De facto, na resolução dos problemas do Grupo IV (grafos com 512 vértices) o algoritmo H_32Itr_10El apresentou um tempo de processamento médio de 1,3 segundos. Já para os problemas do Grupo V (grafos com 1024 vértices) esse valor subiu para 8,1 segundos.

Quanto à razão percentual entre *p* e *n*, apresentada por um problema, parece não ter qualquer efeito sobre o tempo de processamento dos algoritmos guloso, H_1Itr_0El e H_5Itr_3El, como mostra o gráfico da figura 6.6. O mesmo não se pode dizer relativamente ao algoritmo H_32Itr_10El, cujo desempenho parece ser melhor em problemas onde o valor dessa razão é maior.

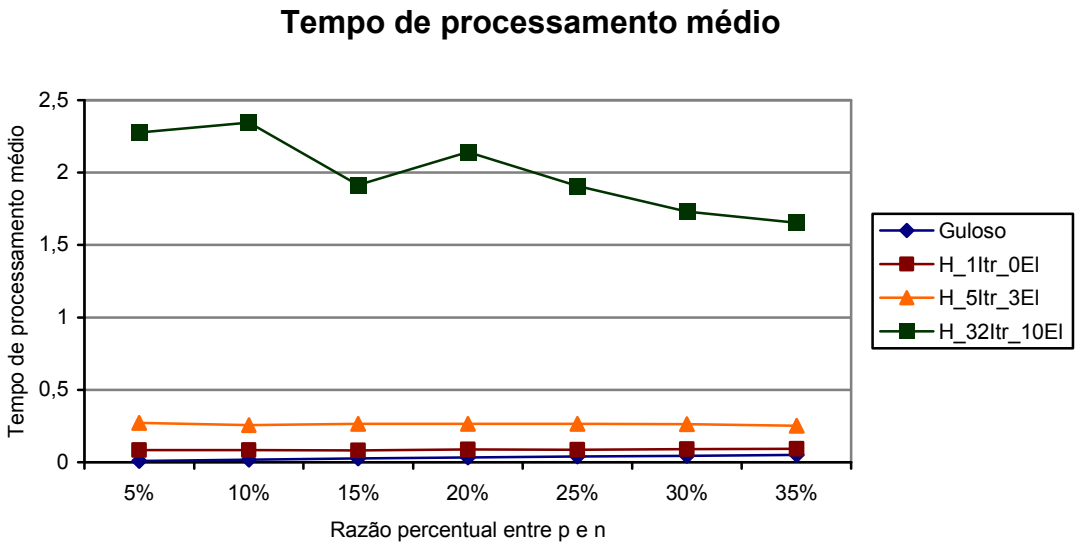

Figura 6.6: Tempo de processamento médio, por algoritmo e tipo de instância.

Balanço entre a qualidade das soluções obtidas e os tempos de processamento

Os resultados dos testes computacionais não deixam margem para dúvidas quer quanto ao algoritmo que obtém soluções de melhor qualidade, o H_32Itr_10El, quer quanto ao algoritmo mais rápido a processar, o guloso. No entanto, quando se aplica um algoritmo na resolução de um problema tão difícil como o problema da gestão óptima da diversidade, espera-se que este obtenha soluções de boa qualidade num espaço de tempo relativamente curto.

A substituição do algoritmo guloso por qualquer uma das versões do híbrido na resolução do problema da gestão óptima da diversidade implicaria um aumento na qualidade das soluções obtidas mas também dos tempos de processamento. No sentido de averiguar qual dos algoritmos testados apresenta o melhor balanço entre a qualidade das

soluções obtidas e os tempos de processamento, apresenta-se, de seguida, um estudo comparativo entre o algoritmo guloso e cada uma das variantes do híbrido. Nesta comparação são tomados como referência os resultados apresentados pelo algoritmo guloso para as instâncias do problema da gestão óptima da diversidade testadas.

Na avaliação da qualidade das soluções será utilizada a medida *percentagem de desvio* ao custo da solução obtida pelo guloso (d_G) cuja definição é a seguinte: dada uma instância, a *percentagem de desvio* ao custo da solução do guloso de uma das variantes do híbrido é a percentagem do custo da solução obtida pelo guloso a que corresponde a diferença entre o custo da solução obtida pelo guloso e o custo da solução obtida pela variante do híbrido, ou seja, $d_G = 100 \times \frac{C_G - C_H}{C_G}$, onde C_G representa o custo da solução obtida pelo guloso e C_H o custo da solução obtida pela variante do híbrido.

Na avaliação dos tempos de processamento obtidos será utilizada a razão entre o tempo de processamento da variante do híbrido e o tempo de processamento do guloso ($\frac{t_H}{t_G}$), onde t_G representa o tempo de processamento do guloso para a instância considerada e t_H o tempo de processamento da variante do híbrido considerada. Note-se que t_G tem de ser diferente de zero, pelo que, nem todas as instâncias geradas puderam ser consideradas neste estudo.

As tabelas B6 a B8 (anexo B) apresentam, para as 71 instâncias onde o tempo de processamento do guloso é diferente de zero, os resultados obtidos para estas medidas.

Guloso vs H_1Itr_0El

O gráfico de dispersão da figura 6.7 mostra a relação existente entre as variáveis d_G e $\frac{t_H}{t_G}$ para cada uma das instâncias consideradas. Note-se que valores maiores para d_G são favoráveis a H_1Itr_0El enquanto que valores maiores de $\frac{t_H}{t_G}$ são favoráveis ao guloso. Por exemplo, o ponto assinalado no gráfico indica que, para aquela instância, o custo da solução obtida pelo H_1Itr_0El é 5% menor do que o da solução obtida pelo guloso. Indica, também, que, nesta instância, o tempo de processamento do H_1Itr_0El é 1,5 vezes maior do que o do algoritmo guloso.

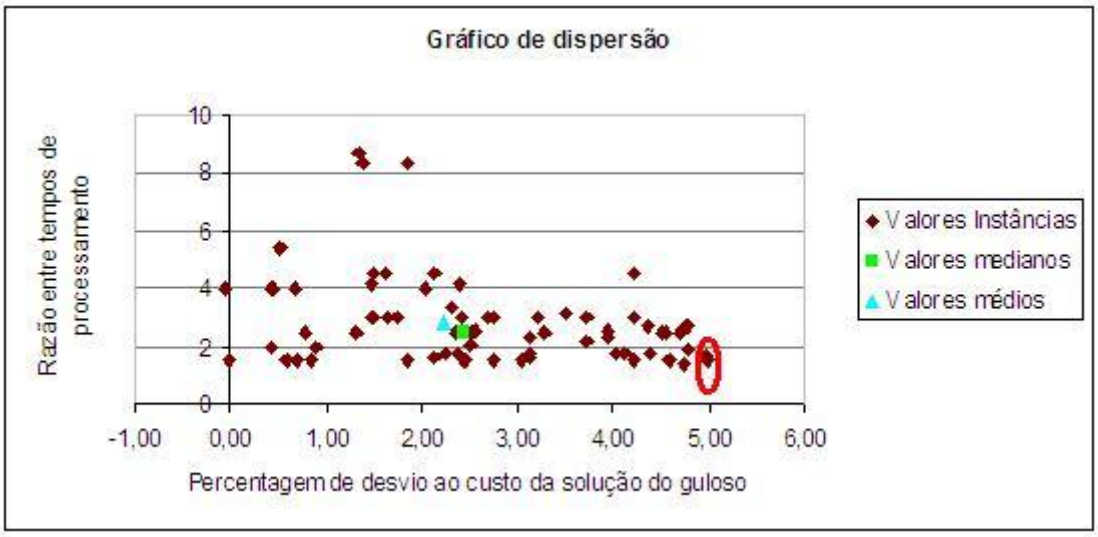

Figura 6.7: Gráfico de dispersão das variáveis d_G e $\frac{t_H}{t_G}$, relativo à versão H_1Itr_0El.

Os resultados mostram que, comparativamente ao guloso, o algoritmo H_1Itr_0El obtém soluções cujo custo é, em média, 2,23% inferior, num tempo de processamento, em média, 2,86 vezes superior.

Os valores medianos das variáveis em estudo indicam que em 50% das instâncias o custo da solução apresentada pelo H_1Itr_0El é pelo menos 2,45% inferior ao custo da obtida pelo guloso. Relativamente aos tempos de processamento, em metade das instâncias consideradas o tempo de processamento do H_1Itr_0El não ultrapassa 2,5 vezes o tempo de processamento do guloso.

Guloso vs H_5Itr_3El

O gráfico da figura 6.8 mostra a dispersão existente entre as variáveis d_G e $\frac{t_H}{t_G}$ para cada uma das 71 instâncias consideradas.

O ponto assinalado no gráfico corresponde a uma instância, na qual o algoritmo H_5Itr_3El obteve um dos seus melhores resultados em termos de balanço entre a qualidade da solução obtida e o tempo de processamento. Nesta instância, o custo da solução obtida pelo H_5Itr_3El é 5,44% inferior ao da solução apresentada pelo guloso e a razão entre os tempos de processamento do H_5Itr_3El e do guloso é igual a 5,5.

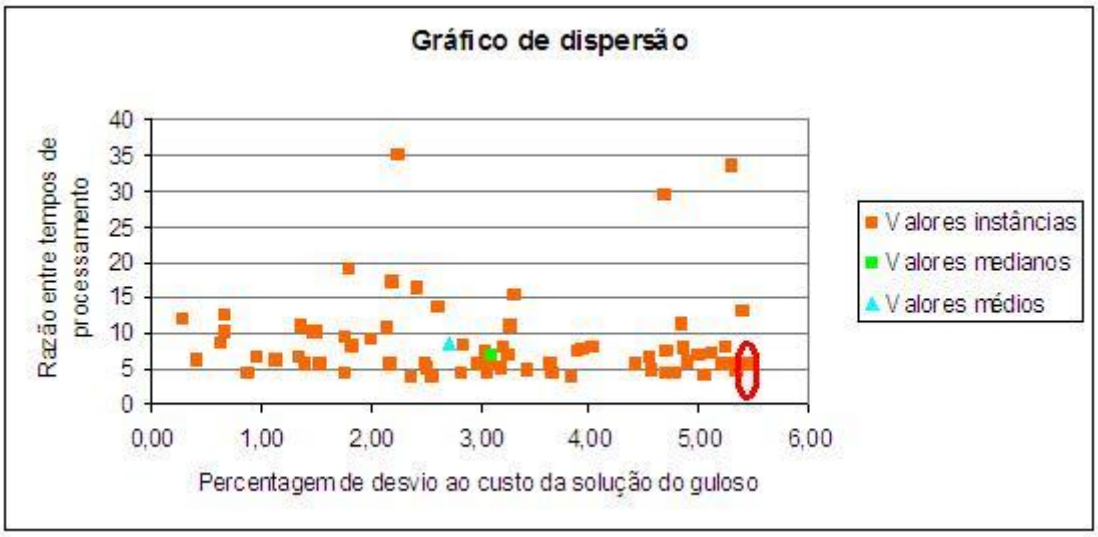

Figura 6.8: Gráfico de dispersão das variáveis d_G e $\frac{t_H}{t_G}$, relativo à versão H_5Itr_3El.

Quando comparado com o algoritmo guloso, o H_5Itr_3El apresenta soluções com um custo 2,73% inferior, em média, e um tempo de processamento, em média, 8,67 vezes superior.

A análise dos valores medianos das variáveis consideradas neste estudo indicam que em metade das instâncias testadas o custo das soluções obtidas pelo algoritmo H_5Itr_3El é pelo menos 3,11% inferior aos custos das obtidas pelo guloso, e que em 50% dos problemas resolvidos o tempo de processamento do H_5Itr_3El é superior a 6,81 vezes o do guloso.

Guloso vs H_32Itr_10El

O gráfico da figura 6.9 mostra a relação existente entre as variáveis d_G e $\frac{t_H}{t_G}$ para cada um dos problemas testados.

A título de exemplo, é assinalado um ponto no gráfico que representa os resultados apresentados pelo algoritmo H_32Itr_10El para uma das instâncias consideradas nos testes. Vê-se claramente que, nesta instância, o algoritmo tem um mau desempenho em termos de balanço entre a qualidade da solução obtida e o tempo de processamento. Embora a solução obtida tenha um custo 2,52% inferior ao custo da solução obtida pelo guloso, o

tempo gasto pelo H_32Itr_10El na obtenção da solução do problema é 420 vezes o tempo de processamento do guloso.

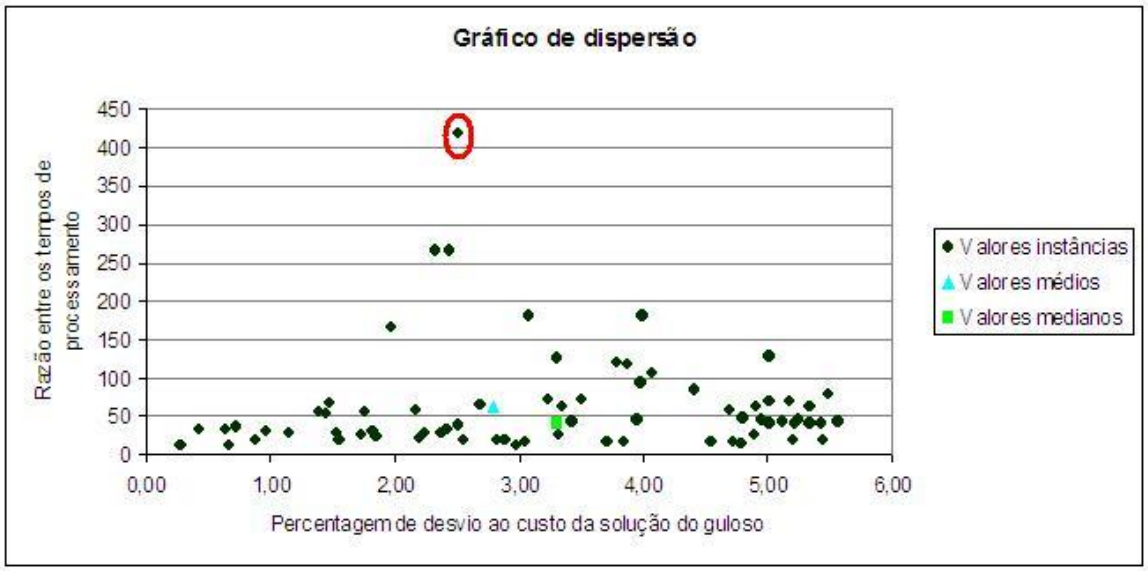

Figura 6.9: Gráfico de dispersão das variáveis d_G e $\frac{t_H}{t_G}$, relativo à versão H_32Itr_10El.

A comparação com o algoritmo guloso mostra que o H_32Itr_10El apresenta soluções com um custo 2,8% inferior, em média, e um tempo de processamento, em média, 62,92 vezes superior.

Os valores medianos indicam que em metade das instâncias testadas o custo das soluções obtidas pelo algoritmo H_32Itr_10El é pelo menos 3,31% inferior aos custos das obtidas pelo guloso, e que em 50% dos casos o tempo de processamento do H_32Itr_10El é superior a 41,95 vezes o do guloso.

6.3. Considerações Finais

Neste capítulo foram apresentados os resultados computacionais obtidos pelos algoritmos guloso e híbrido para 140 instâncias do problema da gestão óptima da diversidade, geradas aleatoriamente, e compararam-se os algoritmos quanto à qualidade

das soluções obtidas, aos dos tempos de processamento e ao balanço entre a qualidade das soluções obtidos e os tempos de processamento.

Nos testes o algoritmo híbrido foi executado com três diferentes conjuntos de valores para os parâmetros de entrada *número de iterações multi-arranque* e *número de soluções elite*: 1 iteração multi-arranque e 0 soluções elite; 5 iterações multi-arranque e 3 soluções elite; e 32 iterações multi-arranque e 10 soluções elite.

Na avaliação da qualidade das soluções obtidas por cada um dos algoritmos foi usada como medida a percentagem de desvio relativo ao custo da solução óptima obtida pelo Xpress. Quando não foi possível encontrar a solução óptima de um problema, considerou-se, na avaliação da qualidade das soluções, a percentagem de desvio ao custo da melhor solução obtida.

No que diz respeito à qualidade das soluções obtidas, os testes confirmaram o que era esperado: o algoritmo H_32Itr_10El obteve soluções mais próximas do óptimo em todas as instâncias.

Relativamente aos tempos de processamento, a análise dos resultados computacionais mostrou que o algoritmo guloso é claramente o mais rápido dos quatro algoritmos comparados, apresentando o menor tempo de processamento em todas as instâncias.

Foi feita uma comparação entre o algoritmo guloso e cada uma das variantes do híbrido em termos de balanço entre a qualidade das soluções obtidas e os tempos de processamento. Nesta comparação foram usadas como medidas a percentagem de desvio do custo da solução obtida pelo híbrido ao custo da solução obtida pelo guloso e a razão entre o tempo de processamento do algoritmo híbrido e o tempo de processamento do guloso. Foram apresentados gráficos de dispersão destas duas medidas que mostram que, numa hipotética substituição do guloso pelo híbrido, na resolução do problema da gestão óptima da diversidade, dever-se-ia recorrer à variante H_1Itr_0El, pois é a mais equilibrada relativamente a dois aspectos essenciais num algoritmo: qualidade da solução obtida e tempo de processamento. Os resultados mostram que tal substituição produziria, em média, uma redução na ordem dos 2,23% nos custos de produção e que o tempo de processamento não atingiria, em média, o triplo do dispendido pelo guloso.

7. Conclusões

Esta tese apresentou o problema da p-mediana e a sua aplicação ao problema da gestão óptima da diversidade.

São diversas as situações reais em que se pretende localizar equipamentos de modo a minimizar a soma das distâncias entre cada ponto de procura e o equipamento localizado mais próximo. É deste tipo de decisão que trata o problema da p-mediana.

Para uma rede geral, o problema da p-mediana é NP-difícil, pelo que, boas soluções podem requerer tempos computacionais excessivos para que possam ser consideradas, por exemplo, no contexto de tomadas de decisão. Isto conduz à necessidade de uma abordagem heurística (ou aproximada) para a resolução do problema.

Foi descrito o algoritmo construtivo guloso, bem como algumas das suas versões aleatorizadas e foram apresentados dois algoritmos de pesquisa local: pesquisa em vizinhança e pesquisa baseada na substituição de vértices. Adicionalmente, foram descritas algumas das metaheurísticas mais utilizadas na resolução do problema da p-mediana: pesquisa tabu, pesquisa em vizinhança variável, algoritmos genéticos e GRASP.

Foi feita a descrição pormenorizada do algoritmo híbrido proposto por Resende e Werneck em [34].

Efectuou-se uma abordagem à aplicação do problema da p-mediana ao problema da gestão óptima da diversidade e apresentou-se a aplicação deste a um problema real da indústria automóvel, denominado *problema da diversidade e distribuição de cablagens*.

Apresentaram-se os resultados obtidos por três versões do algoritmo híbrido e pelo algoritmo construtivo guloso em testes computacionais realizados em 140 instâncias do problema da gestão óptima da diversidade, geradas aleatoriamente. Com base nestes resultados compararam-se os algoritmos testados em termos de qualidade das soluções obtidas, dos tempos de processamento e do balanço entre a qualidade das soluções obtidas e os tempos de processamento. Relativamente à qualidade das soluções obtidas e aos tempos de processamento, os testes confirmaram a superioridade da versão H_32Itr_10El no que diz respeito à qualidade das soluções obtidas e a maior rapidez de processamento

do algoritmo guloso. Quanto ao balanço entre a qualidade das soluções e os tempos de processamento, a versão menos sofisticada do híbrido e o algoritmo guloso destacam-se pela positiva.

Os resultados do estudo computacional sugerem que a substituição do algoritmo guloso pela versão menos sofisticada do híbrido, na resolução das instâncias do problema da gestão óptima da diversidade, implica, em média, 2,9 vezes mais tempo de processamento e uma diminuição de 2,23% no valor dos custos de produção.

Com o objectivo de obter soluções de qualidade semelhante à das soluções apresentadas pela versão menos sofisticada do híbrido num menor tempo de processamento, seria importante, como trabalho futuro, desenvolver um algoritmo que, como o H_1Itr_0El, consistisse numa fase de construção aleatorizada seguida de pesquisa local.

Bibliografia

[1] Agra, A.; Cardoso, D.; Cerdeira, J.; Miranda, M. e Rocha, E.. Solving Huge size instances of the Optimal Diversity Management Problem.

[2] Bresina, J., 1996. Heuristic-biased stochastic sampling. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 271-278, Portland.

[3] Briant, O. e Naddef, D., 2004. The optimal diversity management problem, Operations Research 52(4):515-526.

[4] Briant, O., 2000. Etude théorique et numérique du problème de la gestion de la diversité. Ph.D. thesis, INP Grenoble, France.

[5] Canales, S., 2004. Métodos Heurísticos en Problemas Geométricos: Visibilidad, Iluminacíon y Vigilancia.

[6] Davis, L., 1991. Handbook of Genetic Algorithms., Van Nostrand Reinhold.

[7] Dibbie, C. e Densham, P., 1993. Generating intersecting alternatives in GIS and SDSS using genetic algorithms. GIS/LIS Symposium, Lincoln.

[8] Estévez Valência, P., 1997. Optimización Mediante Algoritmos Genéticos., Anales del Instituto de Ingenieros de Chile, 83-92.

[9] Feo, T. e Resende, M., 1989. A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters, 8:67-71.

[10] Feo, T. e Resende, M., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization, 6:109-133.

[11] García-López, F., Melián Batista, B., Moreno Pérez, J. e Moreno Vega, J., 2002. The parallel variable neighborhood search for the p-median problem. Journal of Heuristics, 8:375-388.

[12] Garey, M. e Johnson, D., 1979. Computers and intractibility: A guide to the theory of NP-completeness, W.H. Freeman and Co., San Francisco.

[13] Glover, F., 1989. Future paths for integer programming and links to artificial intelligence, Computers & Operations Research, 13:533-549.

[14] Glover, F., Laguna, M. e Martií, R., 2000. Fundamentals of scatter search and path relinking. Control and Cybernetics, 39:653-684.

[15] Glover, F., 1996. Tabu search and adaptive memory programming: Advances, applications and challenges. In R. S. Barr, R. V. Helgason, and J. L. Kennington, editors, Interfaces in Computer Science and Operations Research, 1-75. Kluwer.

[16] Goldberg, D.,1989. Genetic Algorithmis in Search, Optimization and Machine Learning. Addison- Wesley.

[17] Hakimi, S., 1965. Optimum distribution of switching centers in a communication network and some related graph theoretic problems, Oper Res 13:462–475.

[18] Hansen, P. e Mladenovic, N., 1997. Variable Neighborhood Search for the p-Median, Location Science 5:207-226.

[19] Hansen, P., 1986. The steepest ascent mildest descent heuristic for combinatorial programming, Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

[20] Hansen, P. e Mladenovic, N., 1999. An introduction to variable neighborhood search. In S. Voß, S. Martello, I. Osman, and C. Roucairol, editors, Meta-heuristics: advances and

trends in local search paradigms for optimization, 30:433.458. Kluwer Academic Publishers.

[21] Hansen, P., e Mladenovic, N., 2001. Variable neighborhood search: Principles and applications. European J. of Oper. Res., 130:449-467.

[22] Holland, J., 1975. Adaptation in Natural and Artificial Systems., University of Michigan Press.

[23] Hosage, C. e Goodchild, M., 1986. Discrete space location-allocation solutions from genetic algorithms. Annals of Operations Research 6:35-46.

[24] Kuehn, A. e Hamburger, M., 1963. A heuristic program for locating warehouses. Management Science 9(4):643-666.

[25] Laguna, M. e Martií, R., 1999. GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS Journal on Computing, 11:44-52.

[26] Maranzana, F., 1964. On the location of supply points to minimize transportation costs. Operations Research Quarterly 12:138-139.

[27] Michalewicz, Z., 1994. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, second edition.

[28] Mladenovic, N., Moreno-Pérez, J. e Moreno-Vega, J., 1995. Tabu search in solving p-facility location-allocation problems, Les Cahiers du GERAD, G:95-38, Montreal.

[29] Moreno-Pérez, J., García-Roda, J. e Moreno-Vega, J., 1994. A parallel genetic algorithm for the discrete p-median problem. Studies in Locational Analysis 7:131-141.

[30] Reese, J., 2006. Solution Methods for the p-Median Problem: An Annotated Bibliography.

[31] Resende, M. e Werneck, R., 2002. A GRASP with path-relinking for the p-median problem. Technical report, Internet and Network Systems Research Center, AT&T Labs Research, Florham Park, NJ.

[32] Resende, M. e Ribeiro, C., 2003. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics, 219-249. Kluwer.

[33] Resende, M. e Werneck, R., 2003. On the implementation of a swap-based local search procedure for the p-median problem. In R. E. Ladner, editor, Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments (ALENEX'03), 119-127. SIAM.

[34] Resende, M. e Werneck, R., 2004. A Hybrid Heuristic for the p-Median Problem.

[35] ReVelle, C. e Swain, R., 1970. Central facilities location, Geographi Anal 2:30–42.

[36] Rolland, E.; Schilling, D. e Current, J., 1996. An efficient tabu search procedure for the p-Median Problem. European Journal of Operational Research 96:329-342.

[37] Teitz, M. e Bart, P., 1968. Heuristic methods for estimating the generalized vertex median of a weighted graph. Operations Research 16:955-961.

[38] Werra, D., 1989. Tabu Search Techniques: A Tutorial and an Application to Neural Networks. OR Spektrum, 11:131-141.

[39] Whitaker, R., 1983. A fast algorithm for the greedy interchange for large-scale clustering and median location problems. INFOR 21:95-108.

Anexo A

Exemplo de resolução do problema da p-mediana numa rede.

De	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
v_1	0	30	74	110	48	120	36	66	96	80	116	134
v_2	30	0	44	80	76	104	66	96	84	110	122	122
v_3	74	44	0	36	32	60	82	56	40	116	78	78
v_4	110	80	36	0	68	24	118	92	48	124	86	68
v_5	48	76	32	68	0	72	50	24	48	94	74	86
v_6	120	104	60	24	72	0	114	84	24	100	62	44
v_7	36	66	82	118	50	114	0	30	90	44	80	122
v_8	66	96	56	92	24	84	30	0	60	74	50	92
v_9	96	84	40	48	48	24	90	60	0	76	38	38
v_{10}	80	110	116	124	94	100	44	74	76	0	38	80
v_{11}	116	122	78	86	74	62	80	50	38	38	0	42
v_{12}	134	122	78	68	86	44	122	92	38	80	42	0

Tabela A1: Matriz de distâncias para a rede da figura 3.2.

Vértice v_i \ Vértice v_j	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
v_1	0	900	2220	3300	1440	3600	1080	1980	2880	2400	3480	4020
v_2	600	0	880	1600	1520	2080	1320	1920	1680	2200	2440	2440
v_3	1776	1056	0	864	768	1440	1968	1344	960	2784	1872	1872
v_4	3960	2880	1296	0	2448	864	4248	3312	1728	4464	3096	2448
v_5	480	760	320	680	0	720	500	240	480	940	740	860
v_6	5760	4992	2880	1152	3456	0	5472	4032	1152	4800	2976	2112
v_7	792	1452	1804	2596	1100	2508	0	660	1980	968	1760	2684
v_8	2112	3072	1792	2944	768	2688	960	0	1920	2368	1600	2944
v_9	2496	2184	1040	1248	1248	624	2340	1560	0	1976	988	988
v_{10}	3520	4840	5104	5456	4136	4400	1936	3256	3344	0	1672	3520
v_{11}	4408	4636	2964	3268	2812	2356	3040	1900	1444	1444	0	1596
v_{12}	5360	4880	3120	2720	3440	1760	4880	3680	1520	3200	1680	0
Total	31264	31652	23420	25828	23136	23040	27744	23884	19088	27544	22304	25484

Tabela A2: Custos para a localização do primeiro equipamento na rede da figura 3.2.

Vértice v_i	Vértice v_j											
	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
v_1	0	900	2220	2880	1440	2880	1080	1980	2880	2400	2880	2880
v_2	600	0	880	1600	1520	1680	1320	1680	1680	1680	1680	1680
v_3	960	960	0	864	768	960	960	960	960	960	960	960
v_4	1728	1728	1296	0	1728	864	1728	1728	1728	1728	1728	1728
v_5	480	480	320	480	0	480	480	240	480	480	480	480
v_6	1152	1152	1152	1152	1152	0	1152	1152	1152	1152	1152	1152
v_7	792	1452	1804	1980	1100	1980	0	660	1980	968	1760	1980
v_8	1920	1920	1792	1920	768	1920	960	0	1920	1920	1600	1920
v_9	0	0	0	0	0	0	0	0	0	0	0	0
v_{10}	3344	3344	3344	3344	3344	3344	1936	3256	3344	0	1672	3344
v_{11}	1444	1444	1444	1444	1444	1444	1444	1444	1444	1444	0	1444
v_{12}	1520	1520	1520	1520	1520	1520	1520	1520	1520	1520	1520	0
Total	13940	14900	15772	17184	14784	17072	12580	14620	19088	14252	15432	17568

Tabela A3: Cálculos para a localização do segundo equipamento na rede da figura 3.2.

Anexo B

Resultados computacionais.

Instância				Guloso			H_1Itr_0El			H_5Itr_3El			H_32Itr_10El		
Nome	n	p	C_O	C	d	t	C	d	t	C	d	t	C	d	t
Grafo1_P1	64	3	4200	4261	1,45	0	4200	0	0,03	4200	0	0,03	4200	0	0,09
Grafo1_P2	64	6	2679	2726	1,76	0	2776	3.61	0,03	2679	0	0,03	2679	0	0,09
Grafo1_P3	64	10	1902	1924	1,18	0	1902	0	0,03	1902	0	0,03	1902	0	0,11
Grafo1_P4	64	13	1511	1532	1,36	0	1524	0,86	0,01	1511	0	0,03	1511	0	0,11
Grafo1_P5	64	16	1225	1238	1,13	0	1232	0,57	0,03	1225	0	0,08	1225	0	0,09
Grafo1_P6	64	19	976	1012	3,73	0	976	0	0,03	976	0	0,03	976	0	0,09
Grafo1_P7	64	22	800	841	5,16	0	800	0	0,02	800	0	0,05	800	0	0,11
Grafo2_P1	64	3	31030	31079	0,16	0	31030	0	0,01	31030	0	0,02	31030	0	0,06
Grafo2_P2	64	6	20452	20785	1,63	0	20452	0	0	20452	0	0,01	20452	0	0,05
Grafo2_P3	64	10	14293	14325	0,22	0	14403	0,77	0	14293	0	0,03	14293	0	0,06
Grafo2_P4	64	13	11208	11710	4,48	0	11254	0,41	0,02	11208	0	0,02	11208	0	0,08
Grafo2_P5	64	16	9194	9637	4,82	0	9233	0,43	0,02	9204	0,11	0,03	9194	0	0,08
Grafo2_P6	64	19	7436	7974	7,24	0	7631	2,62	0	7489	0,72	0,02	7436	0	0,06
Grafo2_P7	64	22	6151	6616	7,57	0	6172	0,34	0,01	6151	0	0,03	6151	0	0,05
Grafo3_P1	64	3	31882	31882	0	0	31882	0	0,01	31882	0	0,02	31882	0	0,08
Grafo3_P2	64	6	20430	20678	1,21	0	20430	0	0	20430	0	0,02	20430	0	0,06
Grafo3_P3	64	10	13600	14177	4,24	0	13599	0	0,02	13599	0	0,02	13599	0	0,08
Grafo3_P4	64	13	10360	10850	4,73	0	10360	0	0,02	10360	0	0,01	10360	0	0,08
Grafo3_P5	64	16	8256	8607	4,24	0	8256	0	0,01	8256	0	0,01	8256	0	0,06
Grafo3_P6	64	19	6647	6996	5,26	0	6647	0	0,02	6647	0	0,01	6647	0	0,06
Grafo3_P7	64	22	5261	5537	5,26	0	5261	0	0	5261	0	0,02	5261	0	0,05
Grafo4_P1	64	3	412525	412525	0	0	412525	0	0,01	412525	0	0,03	412525	0	0,06
Grafo4_P2	64	6	253277	253277	0	0	253277	0	0,02	253277	0	0,02	253277	0	0,05
Grafo4_P3	64	10	144632	144632	0	0	145051	0,29	0	144632	0	0	144632	0	0,06
Grafo4_P4	64	13	103101	103329	0,22	0	103101	0	0,02	103101	0	0,02	103101	0	0,05
Grafo4_P5	64	16	74536	74764	0,31	0	74536	0	0	74536	0	0,02	74536	0	0,05
Grafo4_P6	64	19	62307	62385	0,13	0	62307	0	0,02	62307	0	0,02	62307	0	0,05
Grafo4_P7	64	22	52229	52229	0	0	52229	0	0	52229	0	0,01	52229	0	0,05

Tabela B1: Resultados obtidos para as instâncias do Grupo I: custo da solução, percentagem de desvio, e tempo de processamento em segundos.

Instância				Guloso			H_1Itr_0El			H_5Itr_3El			H_32Itr_10El		
Nome	n	p	C_O	C	d	t	C	d	t	C	d	t	C	d	t
Grafo5_P1	128	6	8173	8272	1,21	0	8173	0	0,02	8173	0	0,03	8173	0	0,17
Grafo5_P2	128	13	5174	5266	1,78	0	5210	0,7	0,01	5200	0,52	0,03	5174	0	0,2
Grafo5_P3	128	19	3810	3984	4,56	0	3810	0	0,01	3810	0	0,06	3810	0	0,2
Grafo5_P4	128	26	2921	3000	2,69	0	2955	1,16	0,03	2921	0	0,03	2921	0	0,17
Grafo5_P5	128	32	2356	2429	3,11	0	2374	0,79	0,03	2356	0	0,03	2356	0	0,17
Grafo5_P6	128	38	1897	1981	4,42	0	1903	0,33	0,03	1897	0	0,01	1897	0	0,14
Grafo5_P7	128	45	1471	1540	4,69	0	1471	0	0,03	1471	0	0,03	1471	0	0,13
Grafo6_P1	128	6	62698	63524	1,32	0	62698	0	0	62698	0	0,03	62698	0	0,14
Grafo6_P2	128	13	40646	41191	1,34	0	41481	2,05	0	40646	0	0,06	40646	0	0,23
Grafo6_P3	128	19	30941	31678	2,38	0	31206	0,86	0,03	30941	0	0,03	30941	0	0,17
Grafo6_P4	128	26	24503	25075	2,33	0	24612	0,44	0,02	24580	0,31	0,05	24503	0	0,22
Grafo6_P5	128	32	20259	20934	3,33	0	20535	1,36	0	20301	0,2	0,03	20259	0	0,19
Grafo6_P6	128	38	16867	17359	2,92	0	16925	0,34	0,02	16867	0	0,05	16867	0	0,17
Grafo6_P7	128	45	13575	14074	3,68	0	13600	0,19	0,03	13574	0	0,03	13574	0	0,13
Grafo7_P1	128	6	39374	39374	0	0	39976	1,53	0,03	39374	0	0,05	39374	0	0,16
Grafo7_P2	128	13	17185	17184	0	0	17184	0	0,03	17184	0	0,03	17184	0	0,11
Grafo7_P3	128	19	9786	9786	0	0	9786	0	0	9786	0	0,03	9786	0	0,13
Grafo7_P4	128	26	6326	6354	0,45	0	6326	0	0,03	6326	0	0,03	6326	0	0,11
Grafo7_P5	128	32	4118	4146	0,68	0	4118	0	0	4118	0	0,01	4118	0	0,09
Grafo7_P6	128	38	3202	3212	0,32	0	3246	1,37	0,02	3202	0	0,03	3202	0	0,11
Grafo7_P7	128	45	2300	2307	0,28	0	2324	1,03	0,02	2300	0	0,05	2300	0	0,13
Grafo8_P1	128	6	641080	697735	8,84	0	650915	1,53	0,02	641080	0	0,03	641080	0	0,14
Grafo8_P2	128	13	384865	406397	5,59	0	384865	0	0,03	384865	0	0,05	384865	0	0,17
Grafo8_P3	128	19	274409	285839	4,17	0	277147	1	0	274409	0	0,05	274409	0	0,14
Grafo8_P4	128	26	189974	199162	4,84	0	190691	0,38	0	189974	0	0,03	189974	0	0,11
Grafo8_P5	128	32	142669	148728	4,25	0	143312	0,45	0,02	142669	0	0,01	142669	0	0,14
Grafo8_P6	128	38	105543	109152	3,42	0	106614	1,02	0,03	105543	0	0,05	105543	0	0,14
Grafo8_P7	128	45	72263	73818	2,15	0	72263	0	0,03	72263	0	0,03	72263	0	0,11

Tabela B2: Resultados obtidos para as instâncias do Grupo II: custo da solução, percentagem de desvio, e tempo de processamento em segundos.

Instância				Guloso			H_1Itr_0El			H_5Itr_3El			H_32Itr_10El		
Nome	n	p	C_O	C	d	t	C	d	t	C	d	t	C	d	t
Grafo9_P1	256	13	14305	14588	1,98	0,02	14395	0,63	0,05	14381	0,53	0,11	14334	0,2	0,52
Grafo9_P2	256	26	9529	9709	1,89	0,02	9641	1,18	0,03	9537	0,08	0,11	9529	0	0,47
Grafo9_P3	256	38	7181	7353	2,4	0	7219	0,53	0,06	7204	0,33	0,13	7181	0	0,42
Grafo9_P4	256	51	5590	5756	2,97	0,02	5615	0,45	0,03	5612	0,39	0,11	5590	0	0,39
Grafo9_P5	256	64	4498	4616	2,63	0,02	4507	0,21	0,05	4498	0	0,09	4498	0	0,37
Grafo9_P6	256	77	3668	3775	2,91	0,02	3705	1,01	0,03	3668	0	0,11	3668	0	0,37
Grafo9_P7	256	90	2986	3078	3,07	0,02	3002	0,53	0,03	2986	0	0,08	2986	0	0,25
Grafo10_P1	256	13	144283	147751	2,4	0	144823	0,37	0,05	144290	0	0,11	144283	0	0,5
Grafo10_P2	256	26	94708	98852	4,37	0	96025	1,39	0,05	95231	0,55	0,11	94708	0	0,56
Grafo10_P3	256	38	72375	76539	5,75	0,02	72945	0,79	0,05	72375	0	0,09	72375	0	0,41
Grafo10_P4	256	51	56854	59977	5,49	0,02	57246	0,69	0,05	56854	0	0,11	56854	0	0,39
Grafo10_P5	256	64	45261	47508	4,96	0,02	45363	0,22	0,05	45275	0,03	0,11	45261	0	0,36
Grafo10_P6	256	77	35886	37692	5,03	0,02	36103	0,6	0,03	35886	0	0,11	35886	0	0,28
Grafo10_P7	256	90	28923	30300	4,76	0,02	29326	1,39	0,06	28923	0	0,09	28923	0	0,33
Grafo11_P1	256	13	91768	94210	2,66	0	91768	0	0,05	91768	0	0,06	91768	0	0,25
Grafo11_P2	256	26	60825	61467	1,06	0	61077	0,41	0,03	60825	0	0,09	60825	0	0,25
Grafo11_P3	256	38	46369	46784	0,9	0,02	46412	0,09	0,05	46369	0	0,09	46369	0	0,39
Grafo11_P4	256	51	35428	35987	1,58	0,02	35679	0,71	0,03	35428	0	0,13	35428	0	0,37
Grafo11_P5	256	64	27670	28544	3,16	0,02	27673	0,01	0,03	27670	0	0,09	27670	0	0,36
Grafo11_P6	256	77	22455	23322	3,86	0,02	22682	1,01	0,03	22467	0,05	0,11	22455	0	0,36
Grafo11_P7	256	90	18333	19066	4	0,02	18438	0,57	0,05	18333	0	0,09	18333	0	0,34
Grafo12_P1	256	13	862020	884205	2,57	0	862020	0	0,03	862020	0	0,11	862020	0	0,33
Grafo12_P2	256	26	498130	509595	2,3	0	498130	0	0,05	498130	0	0,06	498130	0	0,2
Grafo12_P3	256	38	339412	340032	0,18	0	342257	0,84	0,03	339412	0	0,06	339412	0	0,22
Grafo12_P4	256	51	251560	251561	0	0	251560	0	0,03	251560	0	0,06	251560	0	0,22
Grafo12_P5	256	64	195903	195904	0	0	195903	0	0,05	195903	0	0,08	195903	0	0,23
Grafo12_P6	256	77	154228	154663	0,28	0,02	154663	0,28	0,03	154228	0	0,09	154228	0	0,24
Grafo12_P7	256	90	120954	121772	0,68	0,02	121058	0,09	0,03	120954	0	0,08	120954	0	0,22

Tabela B3: Resultados obtidos para as instâncias do Grupo III: custo da solução, percentagem de desvio, e tempo de processamento em segundos.

Instância				Guloso			H_1Itr_0El			H_5Itr_3El			H_32Itr_10El		
Nome	n	p	C_O	C	d	t	C	d	t	C	d	t	C	d	t
Grafo13_P1	512	26	*31030*	32430	4,51	0	31423	1,27	0,08	31071	0,13	0,25	31030	0	2,03
Grafo13_P2	512	51	*21241*	22077	3,94	0,02	21717	2,24	0,09	21404	0,76	0,24	21241	0	2,42
Grafo13_P3	512	77	16006	16685	4,24	0,02	16235	1,43	0,06	16013	0,05	0,25	16006	0	2,13
Grafo13_P4	512	102	12721	13244	4,11	0,03	12751	0,24	0,09	12722	0	0,22	12721	0	1,38
Grafo13_P5	512	128	10143	10503	3,55	0,03	10214	0,7	0,09	10143	0	0,24	10143	0	1,3
Grafo13_P6	512	154	8092	8316	2,77	0,03	8114	0,28	0,09	8097	0,07	0,23	8092	0	1,95
Grafo13_P7	512	179	6447	6669	3,43	0,05	6461	0,2	0,08	6447	0	0,23	6447	0	1,38
Grafo14_P1	512	26	*265249*	268285	1,94	0	266089	1,11	0,09	263177	0	0,27	263177	0	1,61
Grafo14_P2	512	51	*181909*	188045	3,64	0,02	184180	1,51	0,08	181895	0,25	0,27	181449	0	1,47
Grafo14_P3	512	77	*137143*	144201	5,28	0,02	138114	0,84	0,06	137426	0,33	0,31	136970	0	1,42
Grafo14_P4	512	102	109724	115522	5,28	0,02	110659	0,85	0,09	109901	0,16	0,22	109724	0	2,55
Grafo14_P5	512	128	88749	93367	5,2	0,03	89296	0,62	0,08	88834	0,1	0,22	88749	0	1,36
Grafo14_P6	512	154	72925	76853	5,39	0,03	73172	0,34	0,08	72925	0	0,24	72925	0	1,34
Grafo14_P7	512	179	60367	63710	5,54	0,03	60677	0,51	0,08	60367	0	0,23	60367	0	1,35
Grafo15_P1	512	26	114267	114698	0,38	0	115236	0,85	0,09	114267	0	0,2	114267	0	1,22
Grafo15_P2	512	51	64682	65096	0,64	0,02	64809	0,2	0,08	64682	0	0,14	64682	0	0,67
Grafo15_P3	512	77	41340	41518	0,43	0,02	41537	0,48	0,08	41340	0	0,16	41340	0	0,69
Grafo15_P4	512	102	30317	30543	0,75	0,02	30334	0,06	0,08	30334	0,06	0,17	30317	0	0,75
Grafo15_P5	512	128	22015	22270	1,16	0,03	22071	0,25	0,06	22015	0	0,19	22015	0	0,88
Grafo15_P6	512	154	16477	16711	1,42	0,02	16633	0,95	0,08	16483	0,04	0,2	16477	0	1,1
Grafo15_P7	512	179	12169	12359	1,56	0,03	12173	0,04	0,09	12173	0,04	0,19	12169	0	0,87
Grafo16_P1	512	26	1746740	1773082	1,51	0,02	1746778	0	0,06	1746778	0	0,22	1746739	0	1,36
Grafo16_P2	512	51	1115600	1140372	2,22	0,02	1123248	0,69	0,09	1115600	0	0,2	1115600	0	1,17
Grafo16_P3	512	77	796139	810480	1,8	0,02	796455	0,04	0,06	796139	0	0,2	796139	0	1,13
Grafo16_P4	512	102	586734	592482	0,98	0,03	589903	0,54	0,06	586734	0	0,17	586734	0	0,95
Grafo16_P5	512	128	427333	433646	1,48	0,02	430677	0,78	0,08	427781	0,1	0,19	427333	0	1,06
Grafo16_P6	512	154	345435	351873	1,86	0,03	346077	0,19	0,09	345494	0,02	0,2	345435	0	0,95
Grafo16_P7	512	179	282817	289845	2,49	0,03	282817	0	0,09	282817	0	0,2	282817	0	1,03

Tabela B4: Resultados obtidos para as instâncias do Grupo IV: custo da solução, percentagem de desvio, e tempo de processamento em segundos.

Instância				Guloso			H_1Itr_0El			H_5Itr_3El			H_32Itr_10El		
Nome	n	p	C_O	C	d	t	C	d	t	C	d	t	C	d	t
Grafo17_P1	1024	51	*36425*	37160	2,02	0,05	36968	1,49	0,27	36480	0,15	0,95	36425	0	8,31
Grafo17_P2	1024	102	*24700*	25729	4,17	0,06	25179	1,94	0,27	24727	0,11	0,98	24700	0	10,91
Grafo17_P3	1024	154	*18590*	19604	5,45	0,11	18829	1,28	0,28	18624	0,18	0,91	18590	0	7,69
Grafo17_P4	1024	205	*14717*	15573	5,82	0,13	14993	1,88	0,28	14749	0,22	0,98	14717	0	10,41
Grafo17_P5	1024	256	11971	12647	5,65	0,16	12091	1,01	0,28	12007	0,3	1,09	11971	0	10,09
Grafo17_P6	1024	307	9923	10423	5,04	0,17	9993	0,7	0,3	9934	0,11	0,97	9923	0	8,17
Grafo17_P7	1024	358	8248	8672	5,14	0,22	8260	0,15	0,3	8252	0,05	0,94	8248	0	5,89
Grafo18_P1	1024	51	*595164*	610555	2,59	0,03	602071	1,16	0,25	597392	0,37	0,88	595164	0	12,59
Grafo18_P2	1024	102	*405682*	422013	4,03	0,08	412221	1,61	0,27	408447	0,68	0,89	405682	0	9,42
Grafo18_P3	1024	154	*306646*	320826	4,62	0,09	309562	0,95	0,28	306646	0	0,97	306646	0	7,69
Grafo18_P4	1024	205	*242412*	254904	5,15	0,13	244824	0,99	0,3	242425	0,01	1,05	242412	0	8,09
Grafo18_P5	1024	256	195560	207098	5,9	0,16	197175	0,83	0,3	196021	0,24	0,91	195560	0	7,09
Grafo18_P6	1024	307	160518	169588	5,65	0,17	161162	0,4	0,28	160602	0,05	1,01	160518	0	7,09
Grafo18_P7	1024	358	132468	140078	5,74	0,2	133100	0,48	0,3	132501	0,02	0,97	132468	0	8,33
Grafo19_P1	1024	51	*358289*	367235	2,5	0,03	362328	1,13	0,26	360530	0,63	1	358289	0	7,95
Grafo19_P2	1024	102	*234049*	242062	3,42	0,06	236275	0,95	0,25	234532	0,21	0,78	234049	0	7,53
Grafo19_P3	1024	154	171085	176853	3,37	0,11	172352	0,74	0,28	171182	0,06	0,89	171131	0,03	8,09
Grafo19_P4	1024	205	132883	136318	2,59	0,13	132883	0	0,27	132883	0	0,77	132883	0	4,95
Grafo19_P5	1024	256	102611	105115	2,44	0,16	102614	0	0,28	102611	0	0,8	102611	0	4,7
Grafo19_P6	1024	307	77407	79187	2,3	0,17	77413	0,01	0,3	77407	0	0,84	77407	0	5,05
Grafo19_P7	1024	358	56568	57841	2,25	0,19	56606	0,07	0,3	56568	0	0,73	56568	0	3,94
Grafo20_P1	1024	51	*5214230*	5338799	2,39	0,03	5240207	0,5	0,25	5231662	0,33	1,05	5214230	0	7,95
Grafo20_P2	1024	102	*3559815*	3673063	3,18	0,06	3618826	1,66	0,25	3568221	0,24	1,03	3559815	0	10,89
Grafo20_P3	1024	154	*2704725*	2798615	3,47	0,11	2727495	0,84	0,27	2707132	0,09	1	2704725	0	7,05
Grafo20_P4	1024	205	*2162242*	2251811	4,14	0,13	2181560	0,89	0,3	2169886	0,35	1,08	2162242	0	12,28
Grafo20_P5	1024	256	1764730	1852452	4,97	0,16	1794561	1,69	0,28	1767843	0,18	1,09	1765316	0,03	9,48
Grafo20_P6	1024	307	1459040	1536228	5,29	0,17	1474080	1,03	0,3	1461183	0,15	0,95	1459237	0,01	6,89
Grafo20_P7	1024	358	1205976	1272458	5,51	0,2	1214033	0,67	0,31	1206147	0,01	0,95	1205976	0	8,39

Tabela B5: Resultados obtidos para as instâncias do Grupo V: custo da solução, percentagem de desvio, e tempo de processamento em segundos.

Nome da instância	H_1Itr_0El		H_5Itr_3El		H_32Itr_10El	
	d_G	$\frac{t_H}{t_G}$	d_G	$\frac{t_H}{t_G}$	d_G	$\frac{t_H}{t_G}$
Grafo9_P1	1,32	2,5	1,41	5,5	1,74	26
Grafo9_P2	0,70	1,5	1,78	5,5	1,86	23,5
Grafo9_P4	2,45	1,5	2,50	5,5	2,88	19,5
Grafo9_P5	2,36	2,5	2,56	4,5	2,56	18,5
Grafo9_P6	1,84	1,5	2,83	5,5	2,83	18,5
Grafo9_P7	2,46	1,5	2,98	4	2,98	12,5
Grafo10_P3	4,70	2,5	5,44	4,5	5,44	20,5
Grafo10_P4	4,55	2,5	5,21	5,5	5,21	19,5
Grafo10_P5	4,51	2,5	4,70	5,5	4,73	18
Grafo10_P6	4,22	1,5	4,79	5,5	4,79	14
Grafo10_P7	3,21	3	4,54	4,5	4,54	16,5
Grafo11_P3	0,80	2,5	0,89	4,5	0,89	19,5
Grafo11_P4	0,86	1,5	1,55	6,5	1,55	18,5
Grafo11_P5	3,05	1,5	3,06	4,5	3,06	18
Grafo11_P6	2,75	1,5	3,67	5,5	3,72	18
Grafo11_P7	3,29	2,5	3,85	4,5	3,85	17
Grafo12_P6	0,00	1,5	0,28	4,5	0,28	12
Grafo12_P7	0,59	1,5	0,67	4	0,67	11

Tabela B6: Resultados relativos às instâncias do Grupo III.

Nome da instância	H_1Itr_0El		H_5Itr_3El		H_32Itr_10El	
	d_G	$\frac{t_H}{t_G}$	d_G	$\frac{t_H}{t_G}$	d_G	$\frac{t_H}{t_G}$
Grafo13_P2	1,63	4,5	3,05	12	3,79	121
Grafo13_P3	2,70	3	4,03	12,5	4,07	106,5
Grafo13_P4	3,72	3	3,94	7,33	3,95	46
Grafo13_P5	2,75	3	3,43	8	3,43	43,33
Grafo13_P6	2,43	3	2,63	7,67	2,70	65
Grafo13_P7	3,12	1,6	3,32	4,6	3,32	27,6
Grafo14_P2	2,05	4	3,27	13,5	3,51	73,5
Grafo14_P3	4,22	3	4,70	15,5	5,01	71
Grafo14_P4	4,21	4,5	4,87	11	5,02	127,5
Grafo14_P5	4,36	2,67	4,86	7,33	4,95	45,33
Grafo14_P6	4,79	2,67	5,11	8	5,11	44,67
Grafo14_P7	4,76	2,67	5,25	7,67	5,25	45
Grafo15_P2	0,44	4	0,64	7	0,64	33,5
Grafo15_P3	-0,05	4	0,43	8	0,43	34,5
Grafo15_P4	0,68	4	0,68	8,5	0,74	37,5
Grafo15_P5	0,89	2	1,15	6,33	1,15	29,33
Grafo15_P6	0,46	4	1,36	10	1,40	55
Grafo15_P7	1,50	3	1,50	6,33	1,54	29
Grafo16_P1	1,48	3	1,48	11	1,49	68
Grafo16_P2	1,50	4,5	2,17	10	2,17	58,5
Grafo16_P3	1,73	3	1,77	10	1,77	56,5
Grafo16_P4	0,44	2	0,97	5,67	0,97	31,67
Grafo16_P5	0,68	4	1,35	9,5	1,46	53
Grafo16_P6	1,65	3	1,81	6,67	1,83	31,67
Grafo16_P7	2,43	3	2,43	6,67	2,43	34,33

Tabela B7: Resultados relativos às instâncias do Grupo IV

Nome da instância	H_1Itr_0El		H_5Itr_3El		H_32Itr_10El	
	d_G	$\frac{t_H}{t_G}$	d_G	$\frac{t_H}{t_G}$	d_G	$\frac{t_H}{t_G}$
Grafo17_P1	0,52	5,40	1,83	19,00	1,98	166,20
Grafo17_P2	2,14	4,50	3,89	16,33	4,00	181,83
Grafo17_P3	3,95	2,55	5,00	8,27	5,17	69,91
Grafo17_P4	3,72	2,15	5,29	7,54	5,50	80,08
Grafo17_P5	4,39	1,75	5,06	6,81	5,35	63,06
Grafo17_P6	4,13	1,76	4,69	5,71	4,80	48,06
Grafo17_P7	4,74	1,36	4,84	4,27	4,89	26,77
Grafo18_P1	1,39	8,33	2,16	29,33	2,52	419,67
Grafo18_P2	2,32	3,38	3,21	11,13	3,87	117,75
Grafo18_P3	3,51	3,11	4,42	10,78	4,42	85,44
Grafo18_P4	3,95	2,31	4,90	8,08	4,90	62,23
Grafo18_P5	4,79	1,88	5,35	5,69	5,57	44,31
Grafo18_P6	4,97	1,65	5,30	5,94	5,35	41,71
Grafo18_P7	4,98	1,50	5,41	4,85	5,43	41,65
Grafo19_P1	1,34	8,67	1,83	33,33	2,44	265,00
Grafo19_P2	2,39	4,17	3,11	13,00	3,31	125,50
Grafo19_P3	2,54	2,55	3,21	8,09	3,24	73,55
Grafo19_P4	2,52	2,08	2,52	5,92	2,52	38,08
Grafo19_P5	2,38	1,75	2,38	5,00	2,38	29,38
Grafo19_P6	2,24	1,76	2,25	4,94	2,25	29,71
Grafo19_P7	2,14	1,58	2,20	3,84	2,20	20,74
Grafo20_P1	1,85	8,33	2,01	35,00	2,33	265,00
Grafo20_P2	1,48	4,17	2,85	17,17	3,08	181,50
Grafo20_P3	2,54	2,45	3,27	9,09	3,35	64,09
Grafo20_P4	3,12	2,31	3,64	8,31	3,98	94,46
Grafo20_P5	3,13	1,75	4,57	6,81	4,70	59,25
Grafo20_P6	4,05	1,76	4,88	5,59	5,01	40,53
Grafo20_P7	4,59	1,55	5,21	4,75	5,22	41,95

Tabela B8: Resultados relativos às instâncias do Grupo V.